

**Faculty of Engineering of the University of Porto**



# **SLAM for 3D Map Building to be used in a Matching Localization Algorithm**

**Miguel Armando Migueis Pinto**

PhD Thesis

PDEEC- Doctoral Program in Electrical and Computer Engineering

Advisor: Prof. António Paulo Moreira  
Co-Advisor: Prof. Aníbal Coimbra Matos

2012

© Miguel Armando Migueis Pinto, 2012

# Abstract

With a minimum human intervention, tasks can be made safer and the methodical inspection of facilities may become easier.

And, in order to guarantee this minimum human intervention, mobile robots are a solution. Therefore, they must be truly autonomous and able to autonomously pinpoint their location, preferentially without the need to prepare or equip the navigation environment with artificial landmarks or beacons.

This PhD thesis addresses strategies of localization of mobile robots, which can operate without environment preparation. Therefore, and to demonstrate this to be possible, two different approaches in different scenarios were developed.

The first approach, *simple landmark-based* localization system, implemented in a Lego NXT, operates in a constrained and simpler scenario. This approach requires seeing smooth walls in a square shape path, thus enabling vehicle localization and navigation, only within a strict area.

Furthermore, this approach was developed in a suitable way as to be used as a tool to teach concepts of mobile robots to undergraduate students during lectures.

The second approach and scenario, *three-dimensional map-based* localization system, applied to a robotic platform called RobVigil, is much more realistic and has much less constraints. It enables navigation through structured indoor environments, using a 3D map of the building. This last approach focuses on the following two steps:

1) Locating a vehicle, using the state of art Extended Kalman Filter applied to the Simultaneous Localization and Mapping problem (EKF-SLAM), using 2D data as lines for the representation of walls, doors or furniture, and invariant points (invariant with the displacement of the vehicle) representing corners and columns. Once the location has been pinpointed, it will then be possible to build and store the 3D map of the environment, with the help of a rotating Laser Range Finder placed in a servo motor (observation module).

This step is performed within a controlled scenario, without dynamic objects or individuals travelling within the navigation area. It is a preparation, offline and time-consuming task with the purpose of obtaining a 3D map of the facility.

2) Pinpoint the vehicle position during its normal operation, using 3D data and the map obtained in the step previously referred to in 1). The algorithm is now able to compare online (by matching), the observation module readings with the 3D stored map.

The upper side of a building, above the normal height of a person (the headroom), including vertical walls and the ceiling, can be considered a static scenario for large periods of time. Using the static scenario, it is possible to pinpoint the vehicle position during its normal operation in a more robust manner, since there are no dynamic objects or people interfering in the acquired 3D data.

The proposed algorithm improves the computational requirements and execution time, when compared with 2D and 3D SLAM, Particle Filter and algorithms of scan alignment, such as the Iterative Closest Point.

This methodology of localization is discussed in this document as a solution to pinpointing the location of a differential wheel traction robot in dynamic and service environments, with low computational power requirements.

Experimental results on the performance of the proposed method are also presented herein. The comparison between the matching localization algorithm using 2D and 3D data are shown, as well as the advantages of using three-dimensional data.



# Resumo

Existem tarefas que podem ser feitas de um modo seguro e com a mínima intervenção humana, tornando, por exemplo, a inspeção metódica no interior de edifícios mais fácil. Para tal, os robôs móveis são uma solução. Desta forma, estes devem ser verdadeiramente autônomos e capazes de calcular a sua localização no mundo, preferencialmente sem necessidade de se preparar ou equipar o ambiente com marcadores artificiais ou balizas.

Assim, esta tese de doutoramento aborda estratégias de localização, aplicadas a robôs móveis, que operaram sem o auxílio de marcadores artificiais ou balizas. Para demonstrar que tais estratégias são possíveis, duas abordagens e soluções em cenários diferentes foram desenvolvidas.

A primeira solução, localização *simples baseada em marcos naturais* (*simple landmark-based*), foi implementada num Lego NXT, e opera num cenário simples com restrições. Nesta abordagem, o robô necessita de visualizar paredes numa trajetória com forma em quadrado, possibilitando a sua localização e navegação numa área restrita.

Para além disso, foi desenvolvida num modo adequado, visando a sua utilização no ensino, para alunos de graduação durante as aulas de Sistemas Robóticos Autônomos.

A segunda solução, localização *baseada em mapa tridimensional* (*three-dimensional map-based*), é muito mais realista e possui muito menos restrições. Esta, possibilita a navegação em ambientes interiores e estruturados, usando o mapa tridimensional do edifício. Esta abordagem foca-se em duas fases essenciais:

1) Localizar o veículo, usando o estado da arte acerca do Filtro de Kalman Estendido (*Extended Kalman Filter* com acrónimo EKF), aplicado ao problema de localização e mapeamento simultâneo (*Simultaneous Localization and Mapping* com acrónimo SLAM). Para tal, são usados dados bidimensionais, como é o caso de segmentos de reta para a representação de paredes, portas ou mobília; e pontos invariantes (invariantes com o movimento do veículo) representando cantos, esquinas ou colunas. Uma vez obtida a localização do veículo, é possível construir ao mesmo tempo o mapa tridimensional do edifício, com a ajuda de um *tilting Laser Range Finder* (LRF rotativo).

Esta fase é executada com o ambiente controlado, sem objetos dinâmicos ou pessoas a circular dentro da área de navegação do robô. É uma fase de preparação, offline, que consome alguns recursos computacionais, e com o propósito fundamental de obter e guardar o mapa tridimensional do edifício.

2) Obter a posição do veículo durante a sua operação normal, usando dados 3D e o mapa obtido durante a fase anterior. O algoritmo de localização é agora capaz de, online, comparar (*3D Matching*) as leituras do Laser Range Finder rotativo com o mapa 3D guardado. Para tal, são apenas usados dados acerca da parte superior do edifício.

A parte superior de um edifício, acima da altura normal de uma pessoa, incluindo paredes verticais e o teto, pode ser considerado um cenário estático durante longos períodos de tempo. Usando este cenário estático, é possível, a localização do veículo, durante a sua operação normal, de uma forma mais robusta, uma vez que não existem objetos dinâmicos ou pessoas a interferir nos dados 3D obtidos.

O algoritmo de localização proposto é melhor em termos computacionais e tempo de execução necessários, quando comparado com 2D e 3D SLAM, Filtro de Partículas e algoritmos de alinhamento de scanner, como é o caso do *Iterative Closest Point* (com acrónimo ICP).

Esta metodologia, é discutida neste documento, como solução para localizar um robô de tração diferencial, em cenários dinâmicos e de serviços, com baixos requisitos a nível computacional. Resultados experimentais, acerca da performance e precisão da abordagem *three-dimensional map-based*, são apresentados. A vantagem de se usar dados 3D em vez de 2D no algoritmo de *matching* é mostrada.



# Acknowledgments

I would like to thank Prof. António Paulo Moreira and Prof. Aníbal Matos for the advices and the opportunities given so that I could perform my research and studies in the research group ROBIS Group (Robótica e Sistemas Inteligentes), belonging to the INESC TEC Laboratory.

I would also like to express a sincere thank you to my laboratory colleagues. Particularly, I want to thank Heber Sobreira, Marcos Ferreira, Filipe Santos and André Bianchi.

Specially, I want to thank Sandra Costa for believing in me and for the support in all moments.

A special thanks to my family, for their motivation and support. Without them, the conclusion of this PhD Thesis would have been impossible.

Finally, a message for my little niece and goddaughter Carlota, she makes me believe in a good future.





# Official Acknowledgments

This work is funded (or part-funded) by the ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project « FCOMP-01-0202-FEDER-7905».

The author was supported by the Portuguese Foundation for Science and Technology through the PhD grant SFRH/BD/60630/2009.





# Table of Contents

Table of Figures .....	15
Table of Tables .....	23
Table of Algorithms .....	25
Table of Acronyms .....	27
1. Introduction .....	29
1.1. Innovation and Scientific Contributions .....	32
1.2. Document Structure.....	32
2. Literature Review and State of the Art .....	35
2.1. Sensors and Techniques for Localization .....	36
2.1.1. Laser Range Finder.....	39
2.1.2. 3D Observation Module Based on a Laser Range Finder.....	41
2.2. Localization and Mapping Related Work.....	42
2.3. Base Papers and Books.....	50
3. Localization Strategies .....	53
3.1. Pre-Localization and Mapping Modules .....	55
3.2. Localization Module.....	56
4. Simple Landmark-Based Localization.....	59
4.1. Algorithm Implementation .....	60
4.1.1. Control Module .....	61
4.1.2. Association Module .....	62
4.1.3. Observation Module .....	63
4.1.4. Estimation Module .....	66
4.2. Simulation and Real Scenario Communication .....	69
4.3. Application on Education .....	71
4.4. Results on Education .....	73
5. Hardware Setup .....	75
5.1. Robotic Platform .....	75

5.2.	Tilting Laser Range Finder .....	77
5.3.	Odometry .....	83
6.	Data Detection and Extraction.....	87
6.1.	Data Detection.....	87
6.1.1.	Merge and Split .....	89
6.1.2.	Invariant Points Classification.....	90
6.1.3.	Results and Adjustable Parameters.....	92
6.2.	Data Extraction .....	93
6.2.1.	Linear Segment Extraction.....	93
6.2.2.	Circular Segment Extraction .....	98
6.2.3.	Corner Extraction .....	99
7.	Pre-Localization.....	103
7.1.	Filter's State.....	105
7.2.	EKF-SLAM Procedure Cycle .....	111
7.2.1.	Kalman Filter Prediction Stage .....	113
7.2.2.	Association Module .....	115
7.2.3.	Kalman Filter Update Stage .....	118
7.2.4.	Delete Repeated Features Module .....	126
7.3.	EKF-Loc Cycle .....	126
7.4.	Pre-Localization Parameters .....	127
8.	Mapping .....	129
8.1.	Bayes Rules.....	129
8.2.	Creating Distance and Gradient Matrices.....	131
9.	Localization .....	135
9.1.	Three-Dimensional Matching .....	137
9.1.1.	Kalman Filter Prediction.....	137
9.1.2.	3D Perfect Match.....	139
9.1.3.	Kalman Filter Update.....	146
10.	Experimental Results .....	149
10.1.	How to Setup the Localization for Operation in a New Scenario .....	149
10.2.	Pre-Localization Results .....	152
10.3.	Mapping Results .....	160
10.3.1.	3D Occupancy Grid Maps.....	161

10.3.2.	Distance and Gradient Matrices .....	167
10.4.	Localization Results.....	169
10.4.1.	Reference Localization System Sick Nav350 .....	175
10.4.2.	3D Matching Accuracy Characterization.....	177
10.4.3.	Robustness of the Proposed Methodology .....	185
10.4.4.	Execution Time and Comparison with Other Algorithms .....	187
	Conclusions .....	191
	Future Work.....	195
	References .....	197



# Table of Figures

Fig. 1.1 Strategy for building a map in the 3D space, aiming at normal operation using a 3D Matching algorithm. <i>Three-dimensional map-based</i> localization system. ....	31
Fig. 2.1 Kiva robot from Kiva Systems. ....	35
Fig. 2.2 Kiva Systems applied to material handling. ....	35
Fig. 2.3 Cleaning robot Roomba from IRobot. ....	36
Fig. 2.4 Robot Rovio from WowWee. ....	36
Fig. 2.5 Autonomous Underwater Vehicle (AUV) <i>Mares</i> . ....	38
Fig. 2.6 Robotic Soccer Middle Size League (MSL) robot, belonging to the 5DPO team. ....	39
Fig. 2.7 Left: Laser Range Finder-Hokuyo-URG-04LX-UG01. Right: Laser Range Finder Scanning. ....	40
Fig. 2.8. Sick Nav350. ....	41
Fig. 2.9. Cylindrical Reflector used in localization. ....	41
Fig. 2.10 Vehicle location using the MbICP method with a Laser Rang Finder. Image from [38]. ....	44
Fig. 2.11 Left: the raw mapped using odometry. Right: the map generated with the algorithm proposed by S. Thrun and D. Fox [46]. ....	47
Fig. 2.12 The 3D Map built after an accurate vehicle estimation in a 2D space. Image from [46] ....	47
Fig. 2.13 Left: Parent and Child robots. Right: three-dimensional generated map. Images from [48]. ....	47
Fig. 2.14 Reconstructed plans (top). Correspondence with the Hough Transform peaks (bottom). Image form [49]. ....	49
Fig. 3.1 Localization methodology described in the document. ....	54
Fig. 3.2 Corridor, walls in the z coordinate gap of $Z_{min} = 0.2$ metres to $Z_{max} = 1.8$ metres. ....	56
Fig. 3.3 Corridor (projection in a 2D space). ....	56
Fig. 3.4 Vision with a field of view of $360^\circ$ (left), with the detection of green to white transitions. Segmented image on the right: green field (green), field lines (white), obstacles (black) and the ball (orange with circular form). ....	57
Fig. 4.1 LegoFeup. Lego NXT, brick, two IR SHARP sensors, two wheels (differential drive robot). ....	59
Fig. 4.2 Scenario. Left: simulation scenario. Right: real scenario. ....	60
Fig. 4.3 LegoFeup loop. The blue line represents the data flow. The black and bold line shows the process flow. ....	61
Fig. 4.4 Feature association module. ....	62

Fig. 4.5 Real and fitted curve.....	63
Fig. 4.6 Angle observed ( <i>ObsAng</i> ) and distance to the wall ( <i>dw</i> ). ....	64
Fig. 4.7 Vehicle kinematics. Forward distance and rotation travelled, $d$ and $\Delta\theta$ , between consecutive cycle times. ....	67
Fig. 4.8 Algorithm, communication and interface. ....	70
Fig. 4.9 Main Screen - Configuration Tab. ....	70
Fig. 4.10 Main Screen - here, the covariance in x, y and $\theta$ directions are plotted.....	70
Fig. 4.11 Main Screen - here, the vehicle position, angle and covariance ellipse is plotted. ....	70
Fig. 5.1. Left: the RobVigil robot equipped with the observation module. Middle: RobVigil robot in a surveillance task. Right: the robot without covering. The developed <i>observation module</i> can be seen on the top (vehicles head). ....	75
Fig. 5.2. RobVigil robot characteristics: 1.3 meters of height, the tilting LRF on the top with a range of 5 meters ....	76
Fig. 5.3. RobVigil Robot. Its features and sensors. ....	76
Fig. 5.4. The <i>observation module</i> developed.....	77
Fig. 5.5 Laser referential (L). Measurement Referential (D). Point P and reading acquired with the LRF ( $d$ and $\Psi$ ). ....	78
Fig. 5.6 Tilt referential (Tilt) in blue. Laser referential (L) in green. ....	79
Fig. 5.7 Vehicle referential (V) at brown, attached to the vehicle odometry's referential. Tilt referential (Tilt) in blue. ....	79
Fig. 5.8 Vehicle referential (V) in brown. World referential (W) in black. ....	80
Fig. 5.9 Curvature on the readings caused by the AX12-12 rotation.....	82
Fig. 5.10 Linear displacement of each wheel and angular rotation between two different time steps. ....	84
Fig. 6.1 Measures $d \perp$ and $\alpha$ . ....	87
Fig. 6.2 Merge find clusters $ClrPi, f$ and cicular segments $CircPi, f$ .....	89
Fig. 6.3 Split find linear segments $LinPi, f$ ....	90
Fig. 6.4 Invariant points in a squared red and brown shape. The ones which meet the condition 1) are in red. The others which meet the condition 2) are in brown. ....	91
Fig. 6.5 Invariant points classification. The ones which meet the condition 1) are in red. The conditions defined above does not occur in the case of the points $Pfj$ and $Pij + 1$ . ....	91
Fig. 6.6 The Merge and Split method found linear segments (green). Invariant points represented with a green squared shape, while the variant points are represented with a green circled shape.....	93
Fig. 6.7 Application of the Merge and Split. To columns are found and represented in a red circled shape. ....	93
Fig. 6.8 Linear segment referring to the vehicle referential. ....	94
Fig. 6.9 Reading parameters of the observation module. ....	99
Fig. 7.1 Possible strategies (darker grey blocks) for the <i>pre-localization</i> procedure. Left image: <i>pre-localization</i> with a 2D Matching algorithm. Right image: <i>pre-localization</i> with an EKF-SLAM/EKF-Loc algorithm. ....	103



Fig. 7.2 interface between the EKF-SLAM and EKF-Loc procedure and the Data Detection and Extraction Module. At black lines is shown the process sequence, while at blue line is show the data flow. ....	104
Fig. 7.3 Absolute referential ( $x_W, y_W$ ). Vehicle's referential ( $x_V, y_V$ ). Vehicle state $X_v$ . ....	105
Fig. 7.4 Absolute referential ( $x_W, y_W$ ). Vehicle's referential ( $x_V, y_V$ ). Vehicle state $X_v$ . Linear segment representing a wall, with state $XLinjW$ . ....	106
Fig. 7.5 Point j representing a corner, with state $XPntjW$ . ....	107
Fig. 7.6 Point j representing a column, with state $XPntjW$ . ....	107
Fig. 7.7 Example of a inner situation. The inner zone is represented is grey. ....	109
Fig. 7.8 Example of a outer situation. The inner zone is represented is grey. ....	109
Fig. 7.9 Association modules. Correspondence between the observed features and the estimated map state $XmW$ . Correspondence between the observed features and the list of unmapped features. ....	116
Fig. 7.10 $XObsLiniW$ in red. $XLinjW$ in black. Projection of $XObsLiniW$ in green. ...	117
Fig. 8.1. Beam i. Occupied cell represented in blue (oi). Free cells represented in grey, $f1i \dots fni$ . ....	130
Fig. 9.1 Maps on a corridor where experiments were conducted. Distance Matrix. ....	135
Fig. 9.2 Maps on a corridor where experiments were conducted. Gradient Matrix. Distance variation in direction y. ....	135
Fig. 9.3 Maps on a corridor where experiments were conducted. Gradient Matrix. Distance variation in direction x. ....	136
Fig. 10.1 Linear segment projection on the horizontal need to have at least 0.7 metres. $\beta_{min}$ follows this rule. ....	151
Fig. 10.2 $\gamma_{min}$ and $\gamma_{max}$ representation. ....	151
Fig. 10.3 Architectural plan of the scenario 1). The solid black shows the zone where the <i>pre-localization</i> was applied. 1,2,3 and 4 are the divisions of the map. ....	152
Fig. 10.4 2D map obtained with the application of the EKF-SLAM procedure. A square area of 60 metres x 60 metres. ....	153
Fig. 10.5 EKF-SLAM procedure trajectory Vs EKF-Loc trajectory. ....	153
Fig. 10.6 Map 1. Features:52; points:27 and linear segments:25. 25 metres x 25 metres of square area. ....	154
Fig. 10.7 Map 2. Features:74; points:35 and linear segments:39. Corridor with 25 metres length. ....	154
Fig. 10.8 Map 3. Features:50; points:28 and linear segments:22. Area of $25 \times 25 \text{ metres}^2$ . ....	155
Fig. 10.9 Map 4. Features:75; points:44 and linear segments:31. Corridor with about 50 metres. ....	155
Fig. 10.10 A part of the scenario defined as scenario 2). Features:33, points:19, linear segments:14. Area of 20 metres per 60 metres. ....	155
Fig. 10.11 A part of the scenario defined as scenario 3). Features:37, points:23, linear segments:14. Corridor with 30 metres of length. ....	155

Fig. 10.12 A part of the scenario defined as scenario 3). Features: 53, points: 38, linear segments: 15. Area of 30 metres x 30 metres. ....	155
Fig. 10.13 Architectural plan of the (“pavilhão multiusos de Guimarães”), scenario 4). The rectangle in black solid colour, is representative of the zone where the pre-localization was applied.....	156
Fig. 10.14 2D map obtained with the application of the EKF-SLAM procedure, scenario 4). Features:120; points:53 and linear segments:67. A corridor with a length of 60 metres. .	156
Fig. 10.15 Architectural plan of the INESC TEC building, first floor, scenario 5).....	157
Fig. 10.16 2D map obtained with the application of the EKF-SLAM procedure, scenario 5). Features:81; points:42 and linear segments:39. A corridor with a length of 25 metres. ...	157
Fig. 10.17 Architectural plan of the mapped part in the "Fórum do Mar" fair, inside the Exponor building (Matosinhos), scenario 6). ....	158
Fig. 10.18 2D map obtained with the application of the EKF-SLAM procedure, scenario 6). Features:45; points:21 and linear segments:24. A corridor with a length of 25 metres....	158
Fig. 10.19 Architectural plan of a gallery shopping, inside the enterprise centre "Lionesa" (Porto), scenario 7) The solid black colour was the mapped area.....	158
Fig. 10.20 2D map obtained with the application of the EKF-SLAM procedure, scenario 7). Features:42; points:22 and linear segments:20. Square area of 30 metres x 30 metres. ...	158
Fig. 10.21 Architectural plan of the reitoria lobby of the University Porto, scenario 8). The solid grey colour was the mapped area. ....	159
Fig. 10.22 2D map obtained with the application of the EKF-SLAM procedure, scenario 8). Features:70; points:33 and linear segments:37. Square area of 25 metres x 25 metres. ...	159
Fig. 10.23 Image on the left: mapping with odometry. Image on the right: mapping with the EKF-Loc stage algorithm (pre-localization procedure). ....	160
Fig. 10.24 Location where the tests were conducted. Corridor at the Faculty of Engineering of the University of Porto, Portugal. ....	160
Fig. 10.25 Map grid built using Bayes rules.....	160
Fig. 10.26 3D Occupancy grid of the scenario defined as 1). Different perspectives of view. ....	161
Fig. 10.27 Architectural plan of the scenario defined as 2) Feup teachers’ garage. The black solid colour represents the zones which had been mapped. ....	162
Fig. 10.28 3D Occupancy grid of the scenario defined as 2). With circular columns. Different perspectives of view. ....	162
Fig. 10.29 2D map of the scenario defined as 3) students’ corridor B. The black solid colour represents the zones which had been mapped. ....	163
Fig. 10.30 3D Occupancy grids of the scenario defined as 3). Different perspectives of view. ....	163
Fig. 10.31 3D Occupancy grids of the scenario defined as “pavilhão multiusos de Guimarães”. Different perspectives of view. ....	164
Fig. 10.32 3D Occupancy grids of the scenario defined as 5. Different perspectives of view. ....	165
Fig. 10.33 Occupancy grids of the scenario defined as Exponor building during the "Fórum do Mar", scenario 6). Different perspectives of view. ....	165

Fig. 10.34 3D Occupancy grids of the scenario defined as 7). Different perspectives of view. ....	166
Fig. 10.35 3D Occupancy grids of the scenario defined as reitoria's lobby, scenario 8). Different perspectives of view. ....	166
Fig. 10.36 Distance matrix slice in the height of 1.8 metres. $dmap_{x,y,1.8}$ . ....	167
Fig. 10.37 Gradient matrix slice, in the $x$ direction, to the height of 1.8 metres. $\nabla_x 3D_{x,y,1.8}$ . ....	167
Fig. 10.38 Gradient matrix slice, in the $y$ direction, to the height of 1.8 metres. $\nabla_y 3D_{x,y,1.8}$ . ....	167
Fig. 10.39 Occupancy grid of part of the scenario 2) ....	168
Fig. 10.40 Distance matrix slice in the height of 1.8 metres, $dmap_{x,y,1.8}$ . It is about the map of Fig. 10.39. ....	168
Fig. 10.41 Gradient matrix slice, in the $x$ direction, in the height of 1.8 metres, $\nabla_x 3D_{x,y,1.8}$ . It is about the occupancy grid presented in Fig. 10.39. ....	168
Fig. 10.42 Gradient matrix slice, in the $y$ direction, to the height of 1.8 metres. $\nabla_y 3D_{x,y,1.8}$ , about the occupancy grid presented in Fig. 10.39. ....	168
Fig. 10.43 Occupancy grid of part of the scenario 3).....	168
Fig. 10.44 Distance matrix slice for the height of 1.8 metres, about the occupancy grid presented in Fig. 10.43.....	168
Fig. 10.45 Gradient matrix slice for the $x$ direction, in the height of 1.8 metres. About the occupancy grid of Fig. 10.43. ....	169
Fig. 10.46 Gradient matrix slice for the $y$ direction, in the height of 1.8 metres. About the occupancy grid of Fig. 10.43. ....	169
Fig. 10.47 Occupancy grid of part of the scenario 7) ....	169
Fig. 10.48 Distance matrix slice in the height of 1.8 metres. $dmap_{x,y,1.8}$ , about the occupancy grid of Fig. 10.47. ....	169
Fig. 10.49 Gradient matrix slice for the $x$ direction, in the height of 1.8 metres. About the occupancy grid of Fig. 10.47. ....	169
Fig. 10.50 Gradient matrix slice for the $y$ direction, in the height of 1.8 metres. About the occupancy grid of Fig. 10.47. ....	169
Fig. 10.51 GUI interface "rondas". It shows the architectural plan. The marked waypoints are shown in red. The green line shows the trajectory that it is intended to follow. ....	171
Fig. 10.52 The trajectory of the vehicle, Start-End-Start-End, in the scenario 1). In yellow the correct position (3D Matching). In green the odometry position. In the left image the 3D map is seen overhead. In the right image it is seen underneath. ....	173
Fig. 10.53 The trajectory of the vehicle, Start-1-2-3-4-5-6-End, in the scenario 2). In yellow the 3D Matching's position. In green the odometry. In left, the image is seen overhead. In the right image it is seen underneath. ....	173
Fig. 10.54 The trajectory of the vehicle, Start-1-End with loops at the middle, in the scenario 3). In yellow the 3D Matching's position. In green the odometry. Different perspectives of the trajectory. ....	173
Fig. 10.55 The trajectory of the vehicle, Start-1-End, in the scenario 4). In yellow the 3D Matching's position. In green the odometry. Left: the entire occupancy grid is shown. Right,	

only the bottom part (white, below 1.8 metres) is shown. The red part is used for vehicle localization (3D Matching). .....	174
Fig. 10.56 The vehicles position during the trajectory , Start-1-2-End, in the scenario 5) is shown in yellow, through the localization obtained using the 3D Matching. In green the odometry trajectory. ....	174
Fig. 10.57 The trajectory of the vehicle, Start-1-End, in the scenario 6). In yellow the 3D Matching's position. In green the odometry. Only the bottom part of the occupancy grid (below 1.8 metres) is shown with white points. ....	174
Fig. 10.58 The vehicle's trajectory, Start-1-2-3-4-End, in the scenario 7). In yellow the 3D Matching's position. In green the odometry. Different perspectives of the trajectory. ....	174
Fig. 10.59 The vehicle's trajectory from the label Start to the label End, in the last scenario 8). In yellow the 3D Matching's position. In green the odometry. Different perspectives of the trajectory. ....	175
Fig. 10.60. Vehicle, observation module, Sick Nav350 and reflector. ....	176
Fig. 10.61. First scenario where tests of accuracy were performed. ....	176
Fig. 10.62. Estimated location with reference with the Sick Nav350 position. The vehicle position represented at black is obtained with the Nav350. The blue vehicle trajectory is the estimated location. The green six circles are the artificial landmarks, i.e. reflectors placed by the scenario. The corridor has 25 metres long and 6 metres wide. ....	178
Fig. 10.63 $x$ variable in function to the travelled distance. The vehicle estimated position $xv$ is represented in red, while in blue is shown the Nav350's $x$ position. ....	179
Fig. 10.64 $y$ variable in function to the travelled distance. The vehicle estimated position $yv$ is represented in red, while in blue is shown the Nav350's $y$ position. ....	179
Fig. 10.65 $\theta$ variable in function to the travelled distance. The vehicle estimated position $\theta v$ is represented in red, while in blue is shown the Nav350's $\theta$ position. ....	180
Fig. 10.66 Error curves between the 3D matching localization and the Nav350 positioning system estimation. ....	180
Fig. 10.67. Estimated location compared with the Sick Nav350 . The black line is representative of the Nav350 self location. The blue vehicle trajectory is the estimated location. The green six circles are the reflectors. Area of 25 metres of length and 8 metres of width. ....	180
Fig. 10.68 $x$ variable in function to the travelled distance. The vehicle estimated $xv$ is represented in red, while in blue is shown the Nav350's $x$ value. ....	181
Fig. 10.69 $y$ variable in function to the travelled distance. The vehicle estimated position $yv$ is represented in red, while in blue is shown the Nav350's $y$ position. ....	182
Fig. 10.70 $\theta$ variable in function to the travelled distance. The vehicle estimated position $\theta v$ is represented in red, while in blue is shown the Nav350's $\theta$ position. ....	182
Fig. 10.71 Error curves between the 3D matching localization and the Nav350 positioning system estimation. ....	182
Fig. 10.72. Estimated location with reference with the Sick Nav350 position. The black vehicle position is representative of the Nav350 self location. The blue vehicle trajectory is the estimated location. The green six circles are reflectors placed by the scenario. The corridor has 25 metres long and 8 metres wide. ....	183

Fig. 10.73 $x$ variable in function to the travelled distance. The vehicle estimated position $xv$ is represented in red, while in blue is shown the Nav350's $x$ position. ....	184
Fig. 10.74 $y$ variable in function to the travelled distance. The vehicle estimated position $yv$ is represented in red, while in blue is shown the Nav350's $y$ position. ....	184
Fig. 10.75 $\theta$ variable in function to the travelled distance. The vehicle estimated position $\theta v$ is represented in red, while in blue is shown the Nav350's $\theta$ position. ....	184
Fig. 10.76 Error curves between the 3D matching localization and the Nav350 positioning system estimation. ....	184
Fig. 10.77 2D Matching with a dynamic object appearing (red rectangle). Image on the left: without a dynamic object. Image in the middle: with a person in front of the robot. Image on the right: final matching between the actual reading and the 2D Map. Matching is wrong due to the dynamic object. ....	185
Fig. 10.78. Experiment conducted with a fixed observation module (LRF vertically). Image on the left: The performed 3D matching. Image on the right: The inverse of the quadratic matching error. ....	186
Fig. 10.79. Experiment conducted with the observation module rotating. Image on the left: 3D Matching. Image on the right: The inverse of the quadratic matching. ....	186
Fig. 10.80. 2D map grid obtained using the <i>gmapping</i> package of ROS. ....	189
Fig. 10.81. Execution time of the <i>localization</i> phase (3D Matching algorithm). ....	190



# Table of Tables

Table 4.1 Parameters of the fitting curve, shown in equation (4.2), corresponding to the sensor Sharp IR Range Finder.....	63
Table 4.2 Percentage of correct answers plotted. ....	73
Table 5.1 Result values of the odometry calibration.....	86
Table 6.1 Parameters of the Merge and split method.....	92
Table 6.2 Parameters of the invariant points classification method. ....	92
Table 7.1 Parameters of the update in <i>Pre-localization</i> .....	128
Table 7.2 Parameters of association in <i>Pre-localization</i> .....	128
Table 10.1 How to setup the Localization for a new scenario. Brief description of each phase. ....	150
Table 10.2 Demonstrations and exhaustive tests performed by the robot during events and debugging tests.....	172
Table 10.3 Results of the average and standard deviation of $\varepsilon d$ and $\varepsilon \theta$ . Nav350 localization system referenced to ground truth data.....	177
Table 10.4 Results of the average and standard deviation of $\varepsilon d$ and $\varepsilon \theta$ . Comparison between the 3D matching the Nav350.....	179
Table 10.5 Results of the average and standard deviation of $\varepsilon d$ and $\varepsilon \theta$ . Comparison between the 3D matching and GND.....	179
Table 10.6 Results of the average and standard deviation of $\varepsilon d$ and $\varepsilon \theta$ . Comparison between the 3D matching the Nav350 localization positioning system.....	181
Table 10.7 Results of the average and standard deviation of $\varepsilon d$ and $\varepsilon \theta$ . Comparison between the 3D matching and GND.....	181
Table 10.8 Results of the average and standard deviation of $\varepsilon d$ and $\varepsilon \theta$ . Comparison between the 3D matching the Nav350 localization positioning system.....	183
Table 10.9 Results of the average and standard deviation of $\varepsilon d$ and $\varepsilon \theta$ . Comparison between the 3D matching and GND.....	183
Table 10.10 Results of the average and standard deviation of $\varepsilon d$ and $\varepsilon \theta$ for the <i>amcl</i> package of ROS running with 500 and 100 particles. ....	190





# Table of Algorithms

Algorithm 4.1 Extended Kalman Filter.....	66
Algorithm 7.1 EKF-SLAM cycle. ....	112
Algorithm 8.1 Application of the Bayes Rules over the set of cells belonging to the <i>beam<sub>i</sub></i> . ....	131
Algorithm 8.2 Distance transform applied to the occupancy grid <i>M</i> . Initialization of each cell. Step number 1).....	132
Algorithm 8.3 Distance transform applied to the occupancy grid <i>M</i> . Step number 2)....	133
Algorithm 9.1 3D Matching Localization Cycle. ....	137
Algorithm 9.2 Equations of the Kalman Filter Prediction stage. ....	139
Algorithm 9.3 3D Perfect Match Cycle.....	140
Algorithm 9.4 Resilient Back-Propagation algorithm. For each variable that is intended to estimate, <i>xMatch</i> , <i>yMatch</i> and <i>θMatch</i> , the same algorithm runs in a parallel way. ....	143
Algorithm 9.5 Equations of the Kalman Filter Update stage. ....	148



# Table of Acronyms

AGV	Autonomous Guided Vehicle
CCD	Charge-Coupled Device
CMOS	Complementary metal–Oxide–Semiconductor
DGPS	Differential Global Positioning System
DVL	Doppler Velocity Log
EKF	Extended Kalman Filter
EM	Expectation and Maximization
EKF-Loc	Extended Kalman Filter-Localization
EKF-SLAM	Extended Kalman Filter-Simultaneous Localization and Mapping
GND	Ground Truth
GPS	Global Positioning System
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IR	Infrared Sensor
LBL	Long Baseline
LRF	Laser Range Finder
MbICP	Metric Based Iterative Closest Point
MSL	Middle Size League
RANSAC	RANdom SAmple Consensus
RPROP	Resilient Back-Propagation
SBL	Short Baseline
SLAM	Simultaneous Localization and Mapping
ULBL	Ultra Long Baseline
2D LRF	Two-Dimensional Laser Range Finder
3D LRF	Three-Dimensional Laser Range Finder



# 1. Introduction

The applications of mobile robots have not been sufficiently developed due, among other reasons, to difficulties concerning the localization task. Therefore, to be truly autonomous, they must be able to pinpoint their location inside dynamic environments, without preparation requirements.

One of the fundamental characteristics of autonomy is also the ability of the vehicle to move without limitations, a factor that is often introduced by methods such as beacons, which define a maximum range for the navigation.

The errors which occur while estimating the position of the vehicle, as a result of dead reckoning, are accumulative over time. The localization procedure using the dead reckoning information is incremental and uses the previous estimation of the position. This relative positioning leads to an increase of the error estimation without bounds.

Nevertheless, the dead reckoning sensors, in particular the odometry, are the most common used sensors, to be used and fused as redundant information with more complex sensors and localization techniques.

As stated by David A. Schoenwald, [1], the autonomous unmanned vehicles (AUVs)” (...) need to understand enough about their surroundings so that they can function with minimal or no input from humans. This implies the required sensors to be capable of seeing terrain (...)”.

Sensors that can potentially help the dead reckoning in localization are the Laser Range Finders (LRFs) or the infrared sensors (IR). When compared to artificial vision (cameras) and ultrasound sensors, the LRF and IR offers more accurate readings and depth information without the need for other sensors. Also, they are a good solution for navigating in dark and smoky environments.

With regard to computational requirements, the processing of the artificial vision’s images is clearly high when compared with the treatment of the LRF or IR data. Besides, the data rate of sensors as LRF or IR is higher than the artificial vision.

Sometimes, it is not possible or convenient to prepare or equip the indoor environment, intended for vehicle navigation, with any kind of artificial landmarks or features, enabling an easier localization. In addition to the need of preparing the indoor environment, the use of active or passive beacons restricts the vehicle movements within the area covered by the beacon.

The application of localization systems, which require the preparation of the indoor building and a certain setup time, in some situations, became impractical both in aesthetic and functional terms. Furthermore, the costs associated with the preparation of the environment cannot, most of the times, be considered negligible.

Therefore, it was convenient the development of indoor localization strategies, robust, reliable with the ability to correctly perform its function without the need for artificial landmarks or beacons.

In order to fulfil this definition of autonomy, the fundamental goal, motivation and opportunity of this work, described in this PhD document, is the study and implementation of strategies of localization, which solve the problem of cumulative error when odometry is used, without the need for changing or preparing the environment with artificial landmarks or beacons.

In this PhD thesis, first a *simple landmark-based* localization system is described, which uses smooth walls of a room for an efficient localization. This approach is only applicable in simple scenarios, when the vehicle navigates every time inside the same room, using the room's walls, with IR sensors above any interference (near the room ceiling).

Therefore, afterwards, the research of this PhD thesis evolved to fulfil new and more complex requirements as a localization methodology capable of performing its function without the preparation of the environment, using the building infrastructure, without constraints and restrictions in terms of the navigation area, in more complex scenarios. This second approach is called, from now on, *three-dimensional map-based* localization.

Concerning the *simple landmark -based* approach, the goals are the following:

- 1) Implement the approach in a user-friendly robotic platform, with a reasonable price as, for instance, the Lego NXT.
- 2) This approach can be then considered a suitable platform to be used in the teaching of mobile robot concepts to undergraduate students.

Regarding the *three-dimensional map-based* approach, the fundamental goals are:

- 1) Make it possible for the vehicle to locate itself in an indoor environment, with the desired accuracy and autonomy.
- 2) The approach should operate efficiently without the need to prepare the environment with artificial landmarks or beacons.
- 3) The localization algorithm should be able to work correctly in dynamic and structured scenarios, such as: service scenarios, hospitals and department stores.
- 4) The localization methodology should operate with low computational requirements, in computers with low computational power processors.
- 5) The localization algorithm should work in any lighting conditions - dark, window sunlight or smoky environments.
- 6) The approach should be economically viable, to be used as a localization system in a robotic platform.

The first approach, *simple landmark-based*, only allows locating a robot in simple scenarios, with really restrictive constraints. A detailed explanation is presented in Chapter 4.

The same algorithm was used to teach mobile robots concepts, to undergraduate students of the Faculty of Engineering of the University of Porto, Portugal. Aiming to improve the undergraduate students' knowledge and learning during the lectures on Autonomous Robotic Systems, the algorithm was implemented in a Lego NXT, since it is the ideal tool to be used in teaching, because it is an equipment easy to use, at a reasonable price.

At the webpage [2] there are videos and downloads on the *simple landmark-based* approach.

The *three-dimensional map-based* approach is composed by the fundamental tasks: *pre-localization*, *mapping* and *localization*. The entire localization methodology is presented in Chapters 5 to 9.

It is applied to the RobVigil robot, described in Chapter 5. This is a surveillance robot, which navigates inside public facilities, i.e. dynamic environments, as department stores, hospitals (service scenarios), with high degree of autonomy.

The robot RobVigil is a differential wheel traction vehicle, equipped with odometers and a Laser Range Finder, which, by using a servo DC motor, makes it possible to rotate in the tilt position, providing three-dimensional data. In this document, the rotating LRF solution developed, which uses a servo dc motor, will be referred to as the *observation module*.

This localization methodology is sufficiently robust to be used in dynamic scenarios, autonomously follow inspection routes, without being affected by individuals or objects travelling in the navigation area, without environment preparation.

The *three-dimensional map-based* approach uses the Simultaneous Localization and Mapping, to build a 3D occupancy grid of the navigation area, which is used later on, in a fast and online 3D Matching localization algorithm, during the robot normal operation. Therefore, this solution combines SLAM and Matching algorithms, using the best of both.

The self-localization method *three-dimensional map-based*, presented in this document, focuses on the following two steps (see Fig. 1.1):

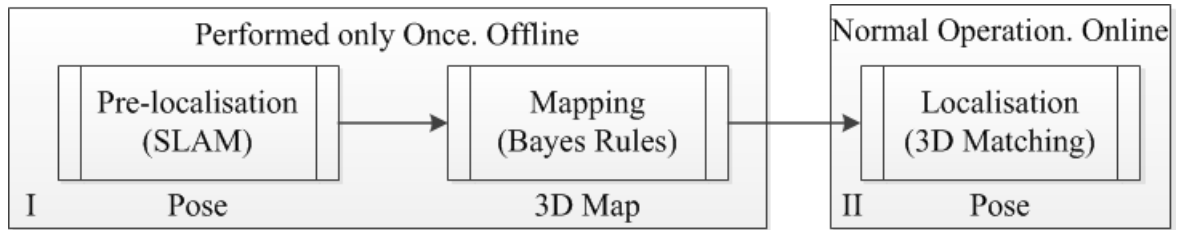


Fig. 1.1 Strategy for building a map in the 3D space, aiming at normal operation using a 3D Matching algorithm. *Three-dimensional map-based* localization system.

I. Locating the vehicle, using a two-dimensional SLAM algorithm with extracted features through the observation module. Once the location has been pinpointed, it is possible to build and store the 3D map of the surrounding environment. This procedure (localization using 2D SLAM and three-dimensional mapping) is performed in a controlled scenario, without dynamic objects or people moving in the navigation area. This is a preparation task that can be performed offline and executed only once. Throughout the document, the procedure of pinpointing the location of the vehicle using the SLAM will be referred to as *pre-localization*, while the three-dimensional mapping task will be referred to as *mapping*.

II. The stored 3D map makes it possible to pinpoint the location of the vehicle by comparing the 3D map and the observation module's readings. This step is performed during the vehicle's normal operation. The used data are 3D points acquired by the *observation module* on the upper side of a facility, which can be considered almost static (without dynamic objects moving). In this document, this task will be called *localization*.

The *three-dimensional map-based* localization algorithm presented herein, improves computational requirements comparatively to 2D and 3D Simultaneous Localization and Mapping (SLAM) algorithms. Furthermore, the time necessary to locate the robot is also reduced when compared to the Particle Filter and algorithms of matching using the consecutive scan alignment, such as the Iterative Closest Point (ICP) algorithms.

Videos and downloads on the *three-dimensional map-based* localization methodology can be found at [3]. In this PhD document experimental results are presented, which validate the developed algorithms.

### 1.1. Innovation and Scientific Contributions

Regarding the *simple landmark-based* localization system, described in Chapter 4, the fundamental contribution was at the level of teaching. The approach was implemented as a suitable tool to be used in teaching, improving the knowledge and learning on mobile robots concepts, to undergraduate students, at the Faculty of Engineering of the University of Porto.

This approach contributed to the scientific field by making undergraduate students more motivated to learn topics such as multi sensor fusion, using probabilistic methods, such as the Extended Kalman Filter, and more interested in further research on the Mobile Robots topic.

Concerning the *three-dimensional map-based* approach, the contributions can be divided into two groups, the ones related to the *pre-localization and mapping* modules and the others related to the *localization* module.

Regarding the *pre-localization and mapping* module the contributions are:

- 1) Development of a solution for SLAM in 2D, using not only linear segments representing walls, doors or furniture, but also points, representing columns and corners.
- 2) 3D mapping, which uses three-dimensional data, based on the 2D SLAM, which uses two-dimensional data. The 3D map is used during the normal operation of the vehicle.
- 3) The *pre-localization and mapping* procedures are performed at the same time using a unique *observation module*, based on a Laser Range Finder.

Regarding the *localization* module, the contributions are:

- 1) Adaptation of a light computational matching algorithm described in [4] by M. Lauren *et al.*, from 2D to the 3D space, using Laser Range Finder data, maintaining low computational requirements.
- 2) Improvement of the fusion system between the matching algorithm described in [4] by M. Lauren *et al.* and odometry data, by using an Extended Kalman Filter.
- 3) Only 3D data on the upper side of a building (almost a static scenario) is used, making localization more robust and reliable, when applied to dynamic scenarios.
- 4) The localization methodology can be used with any observation module which acquires 3D data: Kinect, 3D camera (MESA), stereo vision or commercial 3D LRF.
- 5) Development of a localization methodology that makes the robot RobVigil an economically practicable robotic platform.

### 1.2. Document Structure

The document structure can be divided as follows: Chapter 2 presents a revision on the state of the art related to this work.



In the Chapter 3, the two localization strategies developed in this PhD thesis are introduced: the *simple landmark-based* and *three-dimensional map-based* approaches.

In the Chapter 4 the *simple landmark-based* approach is described. The *three-dimensional map-based* localization system, including the entire set of associated modules, are described from Chapters 5 to 9.

Chapter 5 describes the robot platform, where the tests on the *three-dimensional map* approach, presented in this document, were performed. In this Chapter the robot called *RobVigil* is described in detail, and so are the *observation module* characteristics.

In Chapter 6 the data detection and extraction algorithm is presented. This algorithm outputs 2D features through the data acquired with the *observation module*, which are used during the *pre-localization* procedure.

Chapter 7 describes with detail the *pre-localization* process, performed only once, offline, as a preparation task.

The pose of the vehicle obtained during the *pre-localization* process, described in Chapter 7 allows building an occupancy grid representing the environment in the three dimensional space. This mapping is described in Chapter 8.

Chapter 9 presents in detail the 3D matching algorithm of localization used during the *localization* process, online, during the vehicle normal operation.

In Chapter 10, the results on the entire localization methodology of the *three-dimensional map-based* are presented. Also, an industrial positioning system was used as ground truth aiming to validate the *localization* procedure in different scenarios.



## 2. Literature Review and State of the Art

The need for autonomous vehicles has been growing in society, in the military, in industry or in tertiary services, in the last few years. As a result, it is important to provide mobile robots with a higher level of autonomy.

A higher level of autonomy implies equipping mobile robots with sufficient and helpful intelligence, enabling their perception on the surrounding environment and action upon it.

In modern flexible production systems [5], material handling, storage and services can be performed by Autonomous Guided Vehicles (AGVs). The efficiency and effectiveness of production or service systems is influenced by the level of optimisation of the materials' movement within the plant. Therefore, the performance of the automatic transportation system is crucial to achieve a good overall performance in the production system. The document [6] offers an overview of the technologies and efforts for the AGV systems in handling and logistics, in warehouses and manufacturing plants.

Fig. 2.1 shows a Kiva robot belonging to Kiva Systems and applied to material handling inside warehouses, as shown in Fig. 2.2.



Fig. 2.1 Kiva robot from Kiva Systems.



Fig. 2.2 Kiva Systems applied to material handling.

The introduction of AGV systems and mobile robots increased the autonomy in manufacture and services, resulting in less human intervention and more capacity and efficiency throughout the entire service system. The development of autonomous vehicle systems aims to achieve a higher level of autonomy, standardisation and modularity.

In inspection services and surveillance tasks, the success is directly related to the system's level of autonomy. If this inspection services and surveillance tasks are performed with autonomous vehicles, with minimum human intervention, a better performance in the execution of the task is then guaranteed, without "mistakes", and keeping the operator far from hazardous situations.

The modularity of these systems becomes possible more flexibility for technology integration and improves the industrial and service environment's capacity for change.

Also within civil and service contexts, the growth of the autonomy in robotics has enabled the building of robots which are applicable in our everyday life.

The robot Roomba, shown in Fig. 2.3, is a differential drive robot with ultrasound sensors, which autonomously cleans the house floor. The robot Rovio, Fig. 2.4, is an omnidirectional drive robot, whose localization is possible with the use of beacons, allows the monitoring of small indoor environments, as private houses.



Fig. 2.3 Cleaning robot Roomba from IRobot.



Fig. 2.4 Robot Rovio from WowWee.

## 2.1. Sensors and Techniques for Localization

Different sensors and techniques for the localization of vehicles are described in [7]. These sensors and techniques are divided into absolute and relative localization.

Using sensors and techniques of relative localization, the vehicle's position in the world is given by the sum of successive estimated position differences, leading to a position error that grows without bounds.

The dead reckoning is considered relative localization. Using sensors to perform dead reckoning, the calculation of the vehicle's position, at the present time, is based on its previous position, in the sensor's speed and the elapsed time. This type of estimation leads, over time, to an increase of the error in the calculation of the position of the vehicle.

Examples of dead reckoning sensors are: odometry, accelerometers, gyroscopes, inertial navigation sensors (INS) or inertial measurement units (IMU) and Doppler-effect sensors (DVL).

Due to their high frequency rate, the dead reckoning sensors are commonly used as redundant measurements to support and fuse with more complex localization techniques or sensors, which provide increasingly better quantities of information. Examples include infrared sensors, ultrasound sonars, Laser Range Finders, artificial vision and techniques based on passive or active beacons (such as the triangulation or trilateration).

To perform this fusion probabilistic methods are commonly used, such as the family of Kalman Filters (with special emphasis for the Extended and Unscented Kalman Filters) and the Monte Carlo localization algorithms, known generally as Particle Filters.

Among the dead reckoning sensors and techniques, the most used is the odometry. Some of the problems that appear in the odometry, which cause error in the pose estimation to increase over time are: mechanical imperfections, different radius of wheels and poor calibration of odometry.

J. Borenstein and L. Feng [8], states that the errors in the odometry can be divided into: 1) systematic errors and 2) non-systematic errors. Therefore, in that same paper, they conducted an experiment called, the University of Michigan Benchmark (UMBmark), with the goal of measuring and correcting the most relevant sources of error in odometry.

The accelerometer's output is proportional to the velocity variation. Integrating two times the sensor's signal, it is possible to obtain the distance travelled. This processing leads the increasing of the accelerometer's uncertainty, particularly for smaller accelerations, once also the noise is integrated twice.

Most accelerometers perform badly when applied to small accelerations and velocities. An informal study, developed at the University of Michigan, reveals that, when used in small accelerations, accelerometers have a bad signal/noise ratio, [7].

The output of the gyroscope is proportional to the variation of orientation in time. It produces information on the angular velocity of each sensor's axis. However, much like the accelerometer, in order to obtain the angular rotation, the sensor's signal must be integrated, which implies integrating the noise.

Both, the accelerometers' and gyroscopes' outputs, have also the biases, which are integrated as well and are hard to estimate. However, there are accelerometer and gyroscopes with high precision; on the other hand, they are expensive, have larger dimensions and present high energy-consumption.

An inertial measurement unit (IMU) comprises three gyroscopes and three accelerometers. Each accelerometer measures the linear acceleration, while each gyroscope measures the rotation velocity at each axis.

The IMU inherits the problems and qualities from the accelerometer and the gyroscope, but there are high precision IMUs, even though they are expensive. The work presented by B. Barshan *et al.*, [9], describes the application of an Extended Kalman Filter with an IMU, aiming to estimate the mobile robot's location.

The sensors that use the Doppler effect, Doppler Velocity Log sensors (DVL), compute the vehicle's velocity in relation to the floor. This sensor requires the integration of the signal to obtain the distance travelled, as the inertial navigation sensors, [7].

The position of the vehicle, when is used sensors and techniques of absolute localization, is estimated in the world referential at each instant of time. Examples are the attitude sensors and digital compasses. Some methods of absolute localization are based on passive or active beacons, as is example passive reflectors, acoustic beacons LBL (Long baseline), ULBL (Ultra long baseline) and SBL (Short Baseline) or the Global Positioning Systems (GPS or DGPS) [7].

The attitude sensors and digital compass appear as auxiliary elements of the inertial measurement units (IMU). The attitude sensor operation is based on a magnetic field of the earth. Therefore, the compass is easily influenced by iron infrastructures, power lines and vertical disturbances during the vehicle's motion.

The triangulation and trilateration are two of the essential localization techniques based on active or passive beacons.

In the trilateration, the vehicle position is computed after the measurement of the relative distance between the autonomous vehicle and beacons.

In the triangulation, the vehicle is localized, measuring the angle between the vehicle orientation and each beacon, [7] and [10]. In theory, only three beacons are required to localize any vehicle in space, either using trilateration or triangulation. Two Portuguese works that use triangulation as a method of absolute localization for a mobile robot are: [10] and [11].

Examples of active beacons are the acoustic beacons. Measuring the time of flight of acoustic signals, it is possible to identify the distance between the vehicle and the beacon. A. Matos and N. Cruz at the work [12], localize the Autonomous Underwater Vehicle *Mares* in underwater environments using acoustic beacons, see Fig. 2.5.

The autonomous underwater vehicle, shown in Fig. 2.5 belongs to the robotic group ROBIS, of the INESC TEC in Porto, Portugal, in which the author of this thesis is member.



Fig. 2.5 Autonomous Underwater Vehicle (AUV) *Mares*.

Other examples of active beacons are the Global Positioning Systems (GPS) or the cricket sensor developed by MIT, [13]. The cricket sensor uses a combination of radio frequency and ultrasound sensors, to determine the distance of listeners (attached to mobile devices) in relation to their host devices, which are placed on walls or the ceiling, operating as active beacons.

There are also other sensors, with quality and helpful information on the surrounding environment. Some examples of these type of sensors are: ultrasound sensors, infrared sensors, artificial vision and Laser Range Finder.

The ultrasound sensor measures the time of flight of the sound using the propagation speed to compute the distance of objects. The distance obtained is accurate, although it can be noisy, suffer from signal attenuation (obstacles which are capable of absorbing sound) and from diffuse and specular reflection.

A microcontroller generates a signal with a specific frequency, which is transmitted by the sonar's emitter. The time, detected by the sonar's receptor, between the signal emission and the signal arrival is counted and the distance of the obstacle is determined using the value of the velocity of propagation of the sound, [14].

The infrared sensors measure the distance to obstacles. An example of an IR sensor is the Sharp IR Range Finder, which offers high accuracy. It is small and easy to use, has low power consumption and a thin beam width. It is also economical, which is an important feature, [15].

The Sharp IR Range Finder sensor emit infrared light to obstacles and measure the angle of the light that is reflected. The measured angle allows obtaining the distance from the obstacle. There are two types of IR sensors, the ones that give a binary output and are only able to detect the obstacle's existence; and finally, the IR sensors with a analogue output value that is dependent on the proximity of objects.

A camera stores at each capture instant the light information acquired by a receptor. After, the light information of each pixel, is transformed in the digital form, forming an image. The main types of cameras are divided according the receptor used, CCD or CMOS, [16].

A single camera provides only texture information, while stereo vision, two cameras in different points of the space, as the human eyes, obtains depth information as well. Therefore, the use of two cameras allows obtaining 3D instead 2D information. But the process of calibration in the stereo vision is a hard and complex task. Furthermore, the artificial vision is highly influenced with the light conditions.

The robot in Fig. 2.6 uses artificial vision, a camera with a conical mirror, to have a 360° view about the field in the robotic soccer Middle Size League (MSL). This robot is one of the five belonging to the 5DPO team, the MSL's team of the GroundSys Group of the Faculty of Engineering of the University of Porto, [17]. The author of this PhD thesis is a member of this team.



Fig. 2.6 Robotic Soccer Middle Size League (MSL) robot, belonging to the 5DPO team.

Some of the sensors and localizing techniques lack in the huge error on the vehicle's pose estimation or in the absence of helpful and sufficient information to pinpoint the vehicle location. Other sensors are more affected by the indoor environment and light conditions, such as artificial vision. Finally, other techniques, such as triangulation and trilateration, must use beacons; this requires the environment to be prepared [7].

### 2.1.1. Laser Range Finder

A good choice, for localization and navigation is a Laser Range Finder to be used as observation module. Like the sonar, the Laser Range Finder makes it possible to map the environment and detect nearby objects. The advantages are that the Laser Range Finder can measure long distances and quickly, with an extremely narrow beam, with a precision of millimetres. Therefore, the Laser Range Finder is used to compute distances of objects in

order to obtain a high resolution image of the surrounding environment with a relative large range, [18].

The Laser Range Finder operation consists of: after taking a reading, the laser moves itself within a specific fraction from the total angular range, then another reading is taken. In the "dead zone" or "blind region", of the Laser Range Finder, it is not possible to take readings, typically due to the mechanical constraints, see Fig. 2.7.

The Laser Range Finder specifications are: distance precision, angular resolution and frequency. The distance precision is related to the depth measurement and it is typically about 1-3 mm. The angular resolution is the number of steps in the full possible angular gap. This specification normally exceeds 500 steps. The frequency refers to the repetition of the scanning. It is generally more than or equal to 10Hz. The minimum distance that can often be measured is of a few centimetres and the maximum distance can go from one metre to tens of metres, [18].

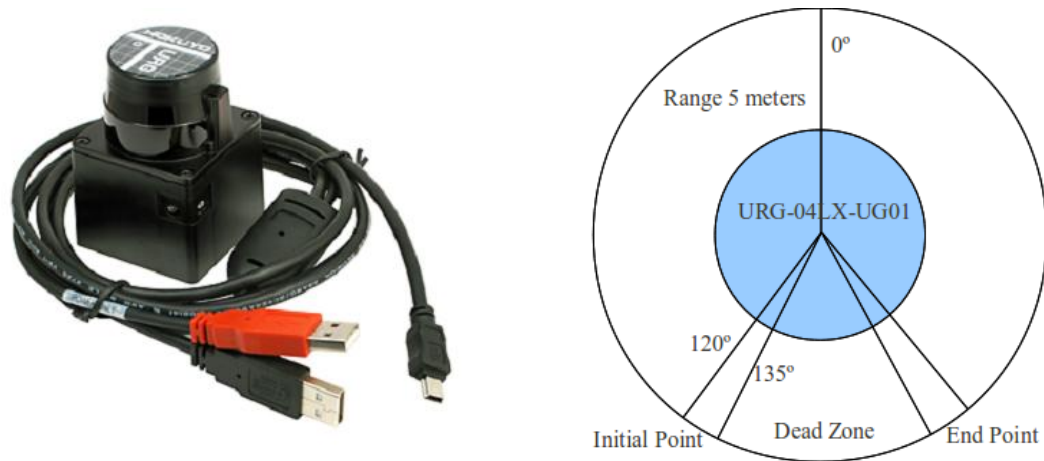


Fig. 2.7 Left: Laser Range Finder-Hokuyo-URG-04LX-UG01. Right: Laser Range Finder Scanning.

Regarding other perception sensors and systems, the complexity and quantity of data obtained by a LRF is smaller when compared to optical vision. However, it is clearly higher when the sensors are ultrasound or infrared sensors.

It is possible to outline some features of the camera and the Laser Range Finder sensors. Regarding the camera sensor: the camera images have a wide range; it has higher resolution; and colour information is available. Regarding the Laser Range Finder: it performs direct acquisition of 2D points; it has no computation overheads; it obtains a large amount of 2D points on surfaces; it has a short acquisition time; light conditions do not affect data acquisition; and finally, it is ideal for the description of irregular surfaces.

The Hokuyo Automatic Co. commercialise cheaper two-dimensional Laser Range Finders with different distance and angular ranges with the robotic purpose.

The SICK Sensor Intelligence is another company which commercialise Laser Range Finder solutions. An example is the Sick Nav350, shown in Fig. 2.8, a laser positioning system. This sensor is commonly applied to autonomous guided vehicles (AGVs). It provides the Laser Range Finder contour and reflectors detection. Some specific scenarios referred by the sensor's developers are: truck loading, shuttle systems and autonomous systems using AGVs.



The sensor is able to detect high reflective reflectors, planar or cylindrical. The reflectors should be covered with a highly reflective film, see Fig. 2.9.

The Sick Nav350 positioning system is capable to provide its self-localization, using a triangulation technique when it detects reflectors and if the reflectors, placed in the navigation area, are mapped and stored in the memory of the sensor.

In addition to its large size, the Sick Nav350 is heavy and expensive, which is impractical to be used in service mobile robots or when the total cost is an important feature.



Fig. 2.8. Sick Nav350.



Fig. 2.9. Cylindrical Reflector used in localization.

### 2.1.2. 3D Observation Module Based on a Laser Range Finder

There are two fundamental types of three-dimensional observation modules. One uses triangulation, making the projection of laser light into a surface. With the help of artificial vision, when the relative position between the source of the laser light and the camera is known, it is possible to reconstruct in the three-dimensional space, the illuminated surface.

This kind of observation module has a limited distance range, but high accuracy, tens of micrometres. A work performed by the author of this PhD thesis on the triangulation topic is described in the paper [19].

The other type uses the time-of-flight concept, i.e. rotating Laser Range Finders to acquire the three-dimensional data. This observation module has an higher distance range, tens of metres, and an accuracy in the order of the millimetres. It is in this last kind of observation modules, the 3D LRF, that this sub-section will focus.

A unique scan do not allow to obtain the 3D representation about the environment. This is only possible with multiple 2D scans pointed in different directions. Therefore commonly, it is used a Laser Range Finder mounted in a movable platform.

There are several three-dimensional commercial sensor solutions capable of perceiving the surrounding environment with high accuracy. Velodyne, Cyra Technologies, Zoller & Frlich, Callidus Precision Systems, SICK Sensor Intelligence, Schmersal EOT, Leica, Mensi, Riegl, OpTech and I-Site are all companies that commercialise 3D laser sensors. However, the price of this equipment still makes it inaccessible for many research groups. Therefore, the absence of a solution with a more accessible price has led some research groups to build their own Laser Range Finder sensors, using cheaper 2D Laser Range Finders placed on movable platforms, [20] to [29].

The 3D LRF can be developed in an economical and feasible way, by taking a 2D Laser Range Finder and a movable or rotating unit. Using the corresponding mathematical transformations, the entire unit (2D LRF plus the movable or rotating platform) will allow to reconstruct the 3D surrounding in a non-expensive way.

D. Klimentjew *et al.* [20] built two different three-dimensional LRF platforms based on the two-dimensional Laser Range Finder URG-04LX. These platforms were tested and approved as economical and feasible alternatives to the commercial three-dimensional Laser Range Finders. The built platforms consisted of: a pan and tilt unit with a LRF, and a robotic arm, with 6 degrees of freedom with the LRF in its end. Both solutions are smaller and lighter than commercial solutions. In terms of price, the built solutions are five to ten times cheaper than the commercial options.

Paulo Dias *et al.* [21], considers that an optimal solution is to use a less expensive 2D Laser Range Finder, creating a sensor that can be used for 3D reconstruction at an affordable price.

A portable 3D LRF, with an angular range of 360°, is presented in the work described by A. Zhang *et al.* [22]. Its cost is about one fifth the price of the commercialised 3D LRFs.

To give the third dimension, the Laser Range Finder can be placed in two different positions: tilt (scanning horizontally) or pan (scanning vertically). Both of these approaches have already been developed by Surman *et al.*, [23] and by Batavia *et al.* [24]. The fundamental difference between the two approaches is in the apex angle orientation and in the influence of moving objects in the reconstructed scenario. In the vertical scanning, the movement of objects and disturbances are less likely to appear between scans. However, the existence of sensed moving objects can be helpful for path-planning and obstacle avoidance, [25].

## 2.2. Localization and Mapping Related Work

There are several works related to the localization of mobile robots using different methods, [10] to [12] and [30] to [33]. However, these works do not follow the research approach proposed in this PhD thesis.

One example is the work of M. Veloso and J. Biswas, described in [33], which uses a prior knowledge on a map grid with Wi-Fi signatures of access points to estimate the location of the vehicle.

The books [34] and [35] are important references in the topics of sensing the surrounding environment, controlling and locating mobile robots.

The algorithms concerning the localization of Mobile Robots can be divided in two large areas: the matching algorithms and the Simultaneous Localization and Mapping algorithms (SLAM), which are capable to solve the problem of localization and mapping at the same time.

In the matching algorithms the pose estimation is commonly fused with dead reckoning data, using for that purpose, probabilistic methods such as the Kalman and Particle Filters.

There are matching algorithms that require prior knowledge on the navigation area. This prior knowledge can be on the environment map, artificial, natural landmarks or beacons.

There are other type of matching algorithms, which compute the overlapping zone between consecutive observations, to obtain the vehicle displacement. One possible matching

algorithm to estimate the quantity of angular and linear displacement of a vehicle between two different and consecutive configurations, is the Iterative Closest Point (ICP). This type of matching, known as point to point matching, analyses the contribution of each point of the laser scan, in the cost function.

The algorithm is composed by two fundamental steps, which are iterated until the solution convergence: the matching and optimising. The result is the distance between consecutive scans, corresponding to the vehicle configuration that minimises the cost function.

The problem of this approach is the amount of data to be processed. The process of finding the correct correspondence between points (matching) is a difficult and time-consuming task.

Examples of works where the scan alignment is used to perform the registration between consecutive scans are the described in the papers [36] to [38].

J. Minguez *et al.* [38], developed the metric-based ICP algorithm (MbICP), improving the standard ICP with a novel distance measure between corresponding points. This measure considers the translation and rotation displacements at the same time, in contrast to the standard ICP. This avoids two different minimisation problems and consequently reduces the computational cost.

Lets Suppose the points  $p_i$  belonging to the previous scan. Lets also imagine that the vehicle displacement is represented by the transformation  $q$ . Therefore, all of the new points ( $p_{inew}$ ) of the actual laser scan can be mapped in the previous laser scan using the  $q$  transformation,  $c_i = q^{-1}(p_{inew})$ . The matching phase, compute the nearest and corresponding point of  $c_i$  in the  $p_i$  set of points. The difference of this step in the MbICP, proposed by J. Minguez *et al.* in [38] is the metric/distance used.

Between consecutive scans, using the least square minimisation error, it is possible to compute the transformation  $q$  that leads to the minimum cost. If the cost function value resulting from the  $q$  transformation is sufficiently small, the convergence of the optimisation step ended. If not, the optimisation step is repeated with the new transformation equal to  $q$ , until reached the convergence end. The first iteration step is performed using the transformation provided by the odometry.

The improvement presented in the metric-based ICP algorithm, makes it possible to achieve a faster convergence to the solution when compared with the standard ICP, with a decreased amount of time spent in each iteration.

J. Minguez *et al.* in [38] conducted an experiment, where the MbICP was applied to a real robot, online in a cycle time of 200 milliseconds, imposed by the Laser Range Finder, which acquires 361 points. The displacement of the vehicle performed during this experiment is shown in Fig. 2.10. The algorithm execution time has not a maximum value and takes a mean of 76 milliseconds in each cycle time, to be executed in a Pentium IV 1.8GHz.

In addition to the computational time spent, the ICP approach has another problem, sometimes there is no sufficient overlapping between two consecutive laser scans and it is hard to find a correct solution. The overlapping zone should obey specific constraints in order to achieve the correct solution. There are shapes, which turn out to be difficult in the registration and can lead to a poor calculation of the vehicle displacement. Some examples are: cones, cylinders, planes and spheres.

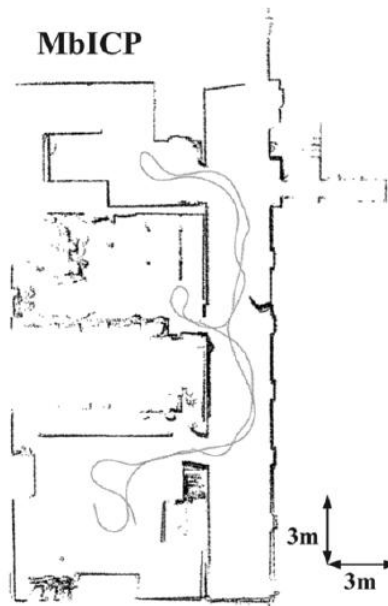


Fig. 2.10 Vehicle location using the MbICP method with a Laser Rang Finder. Image from [38].

The work described by S. Zhang and G. Yan [39], introduces a 3D digitisation method which aims to calculate the next vehicle standpoint. This method guarantees a correct registration with the fewest possible number of overlapping points (to ensure correct matching and reduce the computational time). The approach presented can identify the minimum overlapping area which contains enough information for 3D registration. When the method detects smooth areas, the number of viewpoints increases to obtain larger overlapping zones. In non-smooth areas the number of viewpoints decreases, since a smaller overlapping is sufficient.

In the algorithm described by S. H. Cho and S. Hong in [40], the global map is composed by a set of vertexes (invariant points with the vehicle displacement), representing the environment corners. To reach a specific goal in the space, the vehicle should cross a set of cells. At each grid (point in the map) the vehicle can see a set of vertexes that are helpful to its location.

This work precisely describes the best way of finding and extracting the vertexes. After the matching between the vertexes on the map and the seen in the Laser Range Finder scan, it is possible to directly compute the robot's position in the global referential. Each vertex is represented by a pattern of three elements, the distance to the next vertex, adjacency, which is equal to one if the vertex is connected with the next vertex by a linear segment, zero, if none; and the slant angle between two consecutive vertexes.

This matching algorithm can be described as the following: first the list of vertex is obtained in the laser data; second, the list of vertex is obtained on map (expected vertexes). After, it is tried all the combinations between the two lists of vertexes. Using the pattern of each vertex, a set of conditions need to be obeyed to consider correspondence between vertexes. Finally, the combination between the two lists, which have the larger number of pairs, is used to obtain the vehicle orientation and position.

Some drawbacks of the algorithm proposed by the authors S. H. Cho and S. Hong in [40], are: it is computationally complex to obtain the correspondences between the mapped

vertexes and the ones that are found on the laser data; it is necessary to have a previous map, how can it be obtained?; and finally, it is a complex algorithm to implement.

To some authors, the concept of autonomous self-navigation in mobile robotics can be defined as: true autonomy comes from the capacity that the mobile vehicle has to navigate in unknown environments, building its own global map and simultaneously localizing itself inside the map (Simultaneous Localization and Mapping - SLAM).

Thus, when an environment is well structured, as is example the majority of the indoor environments, it is possible to localize and map the environment at the same time. The most common solutions for the SLAM problem are: the Extended Kalman Filter applied to SLAM, known as EKF-SLAM and the Particle Filter applied to SLAM, known as FastSlam or the Rao-Blackwellized Particle Filter for grid mapping, [41].

A solution for a SLAM problem can build two different types of environment maps: a feature map of artificial or natural landmarks, such as linear segments representing walls, doors or furniture; or an occupancy grid map of occupied or free cells. The feature map is more intuitive and provides more quality and sensible information for human understanding. Furthermore, the feature map has less memory usage when compared with the grid map.

The error in the vehicles' state estimation increases with the vehicle displacement. Each environment feature has an error of estimation (covariance) correlated with the previous mapped features and with the vehicle state. When previous features or map zones are seen in the environment, i.e. a closure loop occurs, the entire map together with the vehicle pose is corrected.

The EKF-SLAM is a variant of the Extended Kalman Filter and uses only one state matrix representing the vehicle state and the landmarks of the feature map. This state matrix is increased every time a new feature is found.

Nonetheless, the EKF-SLAM solution is computationally heavy, quadratic with a number of features  $N$ ,  $O(N^2)$ . In dynamic or large scenarios, as service environments, the map can be increased continuously, thus becoming non-applicable when online performance is required. These aspects can be aggravated if the dimension of the space increases (3D instead 2D space).

On the contrary, the FastSlam solution can be seen as a robot and a collection of  $N$  landmarks estimation problems. Each particle has its pose estimative and tiny state matrices representing each landmarks of the feature map. The FastSlam has a lower computational complexity, when compared with the EKF-SLAM,  $O(M \log N)$ , with  $M$  particles and  $N$  landmarks.

Important works solving the SLAM problem using the EKF-SLAM solution are [42] to [44]. On the contrary, important works that solve the simultaneous localization and mapping problem using the FastSlam approach are [45] to [48].

S. Thrun et al. [46] says that SLAM is the chicken and egg problem. Which came first? the map allowing the vehicle to localize itself or; the vehicle accurate localization, making it possible to map the environment.

At this point, mapping when the position is well known is no longer a problem. This is also true for the opposite scenario; when the mapping is known, the position of the vehicle can be identified. When these two problems are solved at the same time it is called Simultaneous Localization and Mapping, SLAM.

Several localization algorithms accumulate errors during the displacement. In large indoor environments, when a closure loop appears (an area in the navigation environment already visited), they do not have the capacity to close the loop, correcting the entire set of vehicle positions and the map configuration, often acquired with a large amount of error.

Other algorithms based on the Expectation and Maximisation approach (EM) correct the entire map when a closure loop is seen.

The EM family of mapping algorithms consider all past locations, using probabilistic refinement during the map construction, [46]. But, each time an observation arrives, it is impossible in real-time to analyse all of the vehicle's past locations and correct them to produce the actual measure that is consistent with the map. An EM algorithm can take hours, making it impractical for online applications.

S. Thrun and D. Fox [46], developed an algorithm that uses the EM approach, without its entire power, to perform SLAM online and in large cycle environments. Using Particle Filters, samples are distributed around the estimation of position of the vehicle. According to the confidence in the pose estimation, the samples are spaced further apart or closer together. Using a gradient descent maximisation algorithm on each sample, an approximation of the full posterior probability function is obtained. With the full posterior it is possible to obtain the best estimation of the vehicle position and update the map.

Taking the incremental position of the vehicle using the more recent scan and odometry, it is possible to compute other pose estimation. When the difference between the two estimations is different of zero, it means that a shift of the vehicle position has occurred and then a previously mapped area appeared (closure loop). The entire set of vehicle poses in the loop need to be revised, to make the map consistent and increase accuracy in the vehicle pose estimative.

Consequently, the localization in the space is obtained (using a 2D LRF in the horizontal) and the 3D space mapping is possible since the vehicle is well localized (using another 2D LRF, but in the vertical). But, the constructed 3D map is never used to locate the vehicle in its normal operation.

In the following figures, obtained from the work described in [46], it is shown the 2D mapping using SLAM, Fig. 2.11, and the constructed 3D map using another Laser Range Finder, placed vertically, Fig. 2.12.

The publication made by D. Hähnel, W. Burgard and S. Thrun [47], presents an algorithm to generate 3D models of indoor and outdoor environments. To obtain an accurate 3D environment model, a probabilistic scan matching is used to align the present scan with the previous scans. It is used a prior 2D map and it is performed the 2D probabilistic matching. However, as stated by the authors, the three-dimensional variants of the maps and probability functions, consume too much memory. Therefore, this approach is not applicable to 3D scan alignment.

The goal of the work described by T. Yokoya *et al.* [48] is to localize the robot known as the "Parent" robot (which has a 3D LRF module), using a cooperative position system (CPS).



Fig. 2.11 Left: the raw mapped using odometry. Right: the map generated with the algorithm proposed by S. Thrun and D. Fox [46].

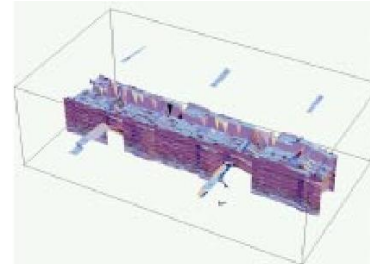


Fig. 2.12 The 3D Map built after an accurate vehicle estimation in a 2D space. Image from [46]

The "Parent" uses the measured relative position to other two robots, called by the authors as "Children". Once the correct localization has been obtained, the "Parent" is able to perform the 3D mapping in indoor or outdoor environments. In other words, the environment scan is only performed when the localization between the "Parent" and "Child" robots is accurate.

The "Parent" and "Children" are shown in following left figure of Fig. 2.13, while the built 3D map is shown at the right figure of Fig. 2.13.

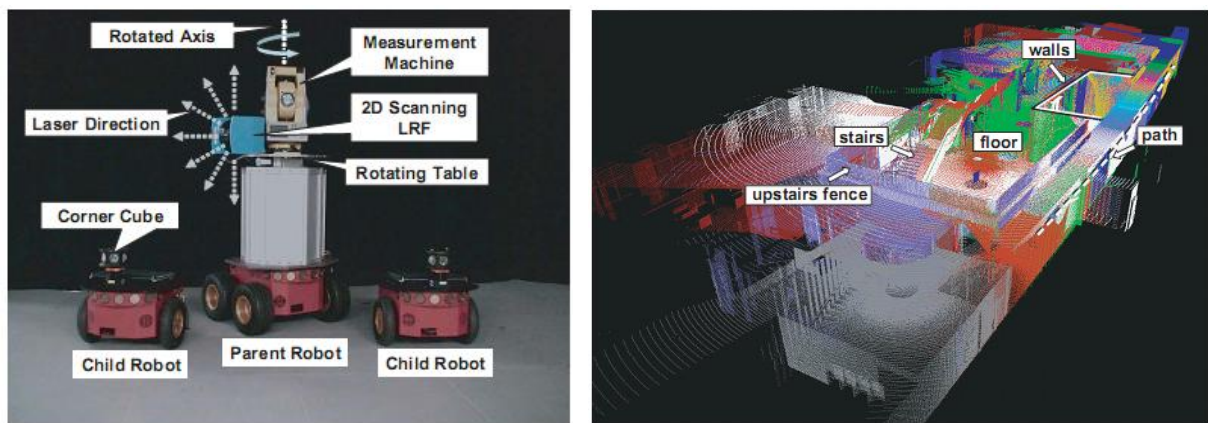


Fig. 2.13 Left: Parent and Child robots. Right: three-dimensional generated map. Images from [48].

Once the initial position of the "Parent" in the world referential is known, the operation of the algorithm is similar to the following steps:

- 1) Once the "Children" has moved, the cooperative position system localizes them in the world referential, computing the position of the each "Child" in relation to the "Parent".
- 2) If the global position of the "Child" is known after the "Parent" displacement, it is possible to know the robot's relative distance and the Parent's new global position.
- 3) When the new global position of the "Parent" is known with a higher level of accuracy, it is possible to perform the LRF scan and accurately map the unknown environment in the three-dimensional space.

An advantage of this work proposed is that it allows transforming the Laser Range Finder data into a 3D map, either in indoor or outdoor environments.

As the three-dimensional LRF is not used for localization, as it is only necessary in the environment mapping, a different method of localization is required (Cooperative Positioning System-CPS). The use of CPS for localization emphasises the need of another method and sensorial module to determine the relative position between vehicles.

Some of the disadvantages of the approach described in the paper are: it requires more than one robot; the cooperative movement is hard to perform, since the "Children" have always to be visible for the "Parent" robot; the path computation is a hard and time-consuming task; and finally, the estimated position of the "Parent" is always a relative estimation, which leads to an increase in the estimation error. Therefore, the "Children" movement need to be as small as possible in order to minimise the error.

T. Yokoya *et al.* [48], provide some values on the time spent during a real experiment, unfortunately, information on the used processor and the number of points acquired by the LRF is not presented. The algorithm is a time-consuming approach: 7 min. to scan the environment and transform the range image into voxel data; 3.5 min. to determine the next position of the "Parent" robot; and 8.5 min. to determine the next position of the "Children". During the experiment: "The Parent robot scanned 25 times and the child robots were displaced 4 times in this experiment. Total time is about 6[hr.]".

The work described by K. Nishimoto *et al.* [49] proves that is possible to use the Hough Transform to solve the SLAM problem or estimate the robot pose. The mapping and localization is based on the extraction of candidates of planes applying a fast Hough Transform.

The localization algorithm presented in this paper can be described as follows:

- 1) Reduction of the Z component, representing the 3D points in a 2D space.
- 2) Applying the Hough Transform in a 2D space, it is possible to find the correct parameters of vertical planes, computing the parameters of lines.
- 3) In the case of slant planes, it is possible to see the points that form a polygon in the 2D projection. It is this set of points that should be tested to determine the plane equation. By using only three of these points, it is possible to obtain the parameterisation of the slant plane in spherical coordinates ( $\rho$ ,  $\gamma$  and  $\theta$ ).

After the vehicle displacement, the same plane in different configurations provides information on its rotation. Between two different configurations it is also possible to know the relation between the  $x$  and  $y$  vehicle coordinates in a 2D space. With two unknown variables ( $x$  and  $y$ ) and only one equation (relation between  $y$  and  $x$ ), another plane is required to estimate the complete vehicle location.

The approach used here reduces the complexity of determining 3D planes using the Hough transform, when 3D data is reduced to the 2D space. Vertical planes are computed using the 2D Hough transform. Slant planes are directly computed when using three points of a polygon.

However, when only planes are used in the localization, a lot of information is lost. A plane may represent, for instance, a wall. A wall has a lot of information that can be helpful in the vehicle localization: windows, holes or corners.

If a plane represents the room's ceiling, the use of its information will not be helpful in the vehicle localization. However, the fire sensors, pipelines or light supports, which there are on the ceiling, can provide a lot of useful information.

Furthermore, other drawbacks appear when using this approach: the Hough Transform is computationally heavy and; since the detected planes and robot position are treated as deterministic variables, whenever a closure point appears (already seen area), the entire map



will not be corrected backwards. Therefore, during the mapping of large areas, the accuracy will decrease.

In this work, the used computer in the calculation of the Hough Transform is Celeron 2.3 GHz. The minimum time required to detect and extract information on the scenario shown at Fig. 2.14 is 45 seconds. In the following figure, Fig. 2.14, it is shown at bottom, the Hough Transform peaks to an particular case, with plans reconstructed on the image of top.

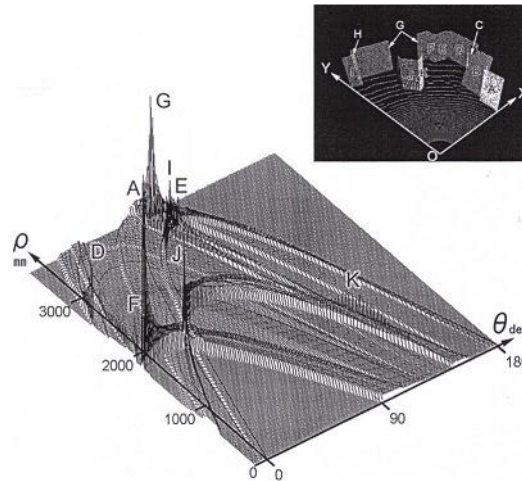


Fig. 2.14 Reconstructed plans (top). Correspondence with the Hough Transform peaks (bottom). Image form [49].

Other works on the topic of mobile robots localization are [50] to [52]. S. Jia *et al.* at the works described in [50] and [51], show the localization and mapping method implemented in a 2D space, based on the combination of two stages: the occupancy grid which is used to construct an map-based on a LRF model and Bayes rules; and the EKF applied to the map is used to correct it and estimate the position of the vehicle.

The occupancy grid step can be described as follows:

- 1) Grid step: each cell is free, occupied or unknown.  $R$  is the distance measured by the Laser Range Finder. The map grid is updated according to the probability of being occupied given a certain observation,  $P(E/O)$  (Bayes rules). If the probability of being occupied is higher than 0.7, then the cell is labelled as occupied.

- 2) Geometry step: remove obstacles and then separate the occupied cells in non-connected zones. Each non-connected zone is then separated into sub-zones, representing a linear segment. The LSQ method is used to obtain the linear segment in each sub-zone.

The second step, the EKF stage, is described as follows:

- 1) Once the local map has been built in the two-dimensional space (linear segments), based on the geometry step, the EKF algorithm is helpful to estimate the robot's pose and correct the map model.

- 2) It is possible to modify the map using an interactive GUI.

The work described by A. Rusdinar *et al.* [52], once again presents a solution for the localization and mapping of mobile robots in the two-dimensional space, using a LRF. But, in the 2D space the LRF readings are affected with the noise introduced by dynamic objects. It describes a Particle Filter that is used to solve the SLAM problem in indoor buildings. The robust mean (weighted average from a small window around the best particle) improves the performance and robustness of the vehicle pose estimation.

The Particle Filter can be divided, fundamentally, into the following three steps: the prediction; the update; and finally the re-sampling.

In the prediction step the particles' state is propagated with the robot kinematics model. During the update step, the weights of the particles are evaluated based on the last Laser Range Finder information. Finally, in the resample step, particles are chosen to survive according to their weight in the filter.

An estimation of the robot's state is obtained by the weighted sum of particles. Some methods of choosing the estimated position are: weight average, the best particle and robust mean. The robust mean is used to compute the position of the vehicle in the work described by A. Rusdinar *et al.* [52]. In this method, a small window centred in the best particle is used. The vehicle pose estimation is the weighted average of all particles in the window.

The use of Particle Filters to estimate the position of the vehicle is a computationally heavy procedure, since the same three basic steps need to be applied to each particle (prediction, update and re-sampling).

In the *Experimental Results* Chapter, sub-section 10.4.4, a deep discussion about the novelties of the *three-dimensional map-based* approach is made, comparing with some works and algorithms of the state of art.

### 2.3. Base Papers and Books

The book "Probabilistic Robotics" from S. Thrun *et al.* [34], is fundamental literature in the mobile robots topic, including localization, sensing, control, multi fusion sensors and resolution of SLAM problems.

Also the book from S. S. Ge and F. L. Lewis [35], gives an important overview about the state of art concerning the mobile robots area. This book, also presents important literature about sensing the environment, control of mobile robots, localization, multi fusion sensors, SLAM applied to indoor and outdoor scenarios and finally, the application of mobile robots.

Both these books, [34] and [35] were fundamental literature in the development of the work presented in this PhD thesis.

The book from A. Gelb *et al.* [53], explains the fundamental theory underlying the Optimal Estimation topic: Kalman Filters, observation error modelling *et cetera*. This book were an important role in the developing work presented in this document.

M. Lauer *et al.* [4] describe a matching algorithm performed in the two-dimensional space with artificial vision, to pinpoint the position of each robot of the Tribots team during the robotic soccer Middle Size League (MSL). This algorithm is called Perfect Match and is executed fast, with low computational power requirements.

This algorithm compares the actual acquired data and the map about the MSL's field to estimate the robot pose in this specific scenario. The Perfect Match was also implemented in the robots of the MSL's team 5DPO, with improvements, where the author is member.

In this paper M. Lauer *et al.* [4], compares the Perfect Match with the Particle Filter algorithm using 200 and 500 particles. The reached conclusion was that, the average spent time when using the Perfect Match, is 4.2 milliseconds, while when using a Particle Filter with 200 particles, the average spent time is about 17.9 milliseconds (four times higher). Furthermore, when using a Particle Filter with 500 particles, the average spent time is about 48.3 milliseconds (ten times higher). The difference of the execution time is really relevant,

especially when is necessary to operate in real-time with high constraints in terms of period cycle. In this experiment, the used processor was the 1GHz Pentium.

As mentioned in Chapter 1, the *Localization* module of the *three-dimensional map-based* approach is based on the paper described by M. Lauer *et al.* [4].

The development of this algorithm in the 5DPO team helped to make clear that this algorithm can be adapted and applied using laser data in the 3D space, performing localization in a simple, robust and non-expensive computational time way.

Successful results such as [54] and [55], attempt to solve the problem of matching between different maps acquired by different robots. If different vehicles are performing the inspection in different zones of a building, the acquired maps should be merged to obtain a larger map. Therefore, different maps are merged without any previous knowledge about their linear and angular offsets. In order to reach this goal, the authors used the Random Walk algorithm to minimise the dissimilarity between the maps in the overlapping zone. The authors present an acceptance indicator that serves as last stage of acceptance for the solution achieved when the algorithm is applied.

In the *three-dimensional map-based* approach, these documents, [54] and [55], are important as they contain the pseudo-code algorithm which is helpful when computing the distance matrices that work as a look-up table during the *localization* procedure.

As well the gradient matrices are look-up tables in the procedure of *localization*. The book from R. C. Gonzalez and R. E. Woods [56], describes techniques for image processing. Among others, the Sobel and Prewitt filters are described in this book, which are helpful to obtain those gradient matrices.

Regarding the *Pre-Localization and Mapping* procedure on the *three-dimensional map-based* approach, the base documents are: [41], [43], [44], [57] to [59].

For the authors of [40], the Kalman and Particle Filters, applied to Simultaneous Localization and Mapping problems (SLAM) fail when false detection of features occurs. In fact, this is one of the greater challenges in the SLAM problem.

However, there are already improvements and solutions for the data association problem in SLAM. Two solutions, which work together, are the individual and joint compatibility between the feature measured and the features that already exist in the map state.

The individual and joint compatibility can use the *Mahalanobis* distance to find the corresponding feature, taking into account the accuracy of the obtained measure and the accuracy of the mapped feature. As is presented at the approach proposed in [41] and [58].

Other approach to associate mapped features with found features in the actual set of readings, is proposed in [43]. In the developed *pre-localization* procedure, it is performed an association between an observed feature and a feature already belonging to the EKF, using an approach based on the work presented by Andrea Garulli *et al.* [43].

In the work described by L. Teslić *et al* [44], the linear segments are parameterised in polar coordinates, such as: distance to the origin  $\rho$  and the angle that the line has in relation to the x-axis ( $\theta$ ). Therefore, each linear segment parameter has noise associated that is modelled as Gaussian noise with a mean of zero and standard deviation  $\sigma_\rho$  and  $\sigma_\theta$ .

In this paper, the linear segment parameters are computed using the classic Least Square Quadratic (LSQ) error method instead the orthogonal LSQ. The paper also describes how to compute the parameters' covariances using the classic LSQ. It essentially focuses on how to

generate output covariance matrices related to the environment lines. The classic LSQ is proved to be more efficient and to take less computational time than the orthogonal LSQ.

The classical LSQ accuracy depends on the number of points while the orthogonal LSQ accuracy depends on the LRF noise model.

L. Teslić *et al.* [44], proved that: the classic LSQ method, used to determine the linear segment parameters and output covariance matrices, has a lower variance. Since in this paper, the linear segments are computed together with the calculation of the respective parameters and the covariance matrix, they become a fundamental part of the work developed in this thesis, aiming to develop the SLAM algorithm using 2D data.

The paper [57], compares methods for line extraction, applied to Laser Range Finder data for indoor mobile robots. The algorithms described and compared are: 1) Split-and-Merge; 2) Incremental Algorithm; 3) Line Regression; 4) Ransac; 5) Hough Transform; and finally 6) Expectation and Maximisation (EM).

Some of the features of the algorithms outlined above are:

1) The Split-and-Merge, Incremental and Line Regression, are much faster than the others; this is essentially due to: a) their nondeterministic nature and b) because they make use of the scan data that is a sequence of raw scan points.

2) The Incremental algorithms are correct, since they have a very low number of false positives. The Split-and-Merge is better in true positive line detections.

3) The Split-and-Merge and Incremental are the preferred methods used for indoor robotic localization and mapping. However, the Split-and-Merge is probably the most popular line extraction algorithm, due to its speed, accuracy and relative precision with the line parameters.

4) RANSAC and EM produce a high number of false positive line detections.

5) Algorithms as RANSAC, Hough Transform and EM are slower, however they produce more precise lines.

The work described in [59] presents a particular and interesting approach to precisely determine the invariant points and the respective errors. The author modelled the error of a corner as a sum of two fundamental causes: 1) measurement process noise, i.e. error in the distance measurement. This error is modelled by the equation:  $e = a \cdot r_i + b$ , where  $a$  is Gaussian noise with zero mean and standard deviation of  $\sigma_a$ .  $b$  is Gaussian noise with zero mean and standard deviation  $\sigma_b$ . 2) Quantisation error, that occurs in the start-end cluster. It is the error that is caused by the digitalisation of the laser scan. It is modelled as the maximum gap between two consecutive distance measurements. It is dependent on the scanner's angular resolution, the distance measured and the angle that the line, which contains the corner, makes with the measurement vector. The angular error is modelled as Gaussian noise with zero mean and standard deviation  $\sigma_\theta$ .

The books and papers enumerated in this Chapter were a fundamental role in the developed work, which is described in this thesis document. These documents are addressed along this thesis, in the parts of respective interest.

### 3. Localization Strategies

As described in Chapter 1, two different localization methodologies were implemented, with the fundamental goal of localize mobile robots without the need of prepare or equip the indoor environment with artificial landmarks or beacons: *simple landmark-based* and *three-dimensional map-based* approaches.

The localization strategies are sufficiently fast to run online in the respective platforms. Both approaches make use of natural landmarks and are applied to the same type of robot, a differential drive robot, but can be used in robots with other types of traction.

Therefore, the kinematic model used in the *simple landmark-based* approach is used as well in the *three-dimensional map-based* approach, even applied to different robots. The association techniques and multi fusion sensor method (Extended Kalman Filter) used in the Lego's localization (*simple landmark-based*), is also used in the *three-dimensional map-based* localization methodology.

The *simple landmark-based* approach is applied to a Lego NXT in a simple and constrained scenario. In this scenario the vehicle need to follow a square shape path and locate itself using walls at each side of the path.

To do that, the Lego NXT uses the measurements of two IR Sharp sensors, to compute the distance and orientation of the robot in relation to those walls, whose positions are previously known.

Furthermore, encoders in each wheel of the Lego NXT, are used to obtain odometry data. The odometry data is fused with the observation of the walls, allowing to compute the estimation of the robot pose.

The strategy used for the Lego NXT and the implementation of this localization algorithm, is explained in detail in Chapter 4. Furthermore, this approach was developed in a suitable way, to be used to teach the Extended Kalman Filter and its application in mobile robots. As well in Chapter 4 experiments with undergraduate students are described.

Although being a localization methodology with potentialities, the *simple landmark-based* approach is only applicable in simple scenarios, as a square shape with walls at each side.

Therefore, this PhD thesis evolved in the sense of develop a suitable localization approach, capable of pinpoint an indoor vehicle in more complex scenarios, without constraints, using for that, only the infrastructure of the building, without artificial landmarks or beacons. As already referred, this approach is called *three-dimensional map-based*.

The *three-dimensional map-based* methodology uses the three-dimensional map of the surrounding environment to pinpoint the robot pose. This approach fulfil the requirements and

solves the problem described in the Chapter 1, performing the localization with the following two steps:

1) *Pre-localization* and *mapping*: to obtain the three-dimensional map of an indoor environment; here it is intended to navigate with a vehicle equipped with an *observation module* capable to acquire three-dimensional data.

2) *Localization*: to perform the vehicle self-localization; this is to be used online using a matching algorithm, in the three-dimensional space, between the present readings and the stored maps.

These two different modules are performed offline and online, respectively, 1) *pre-localization and mapping*; and 2) *localization*.

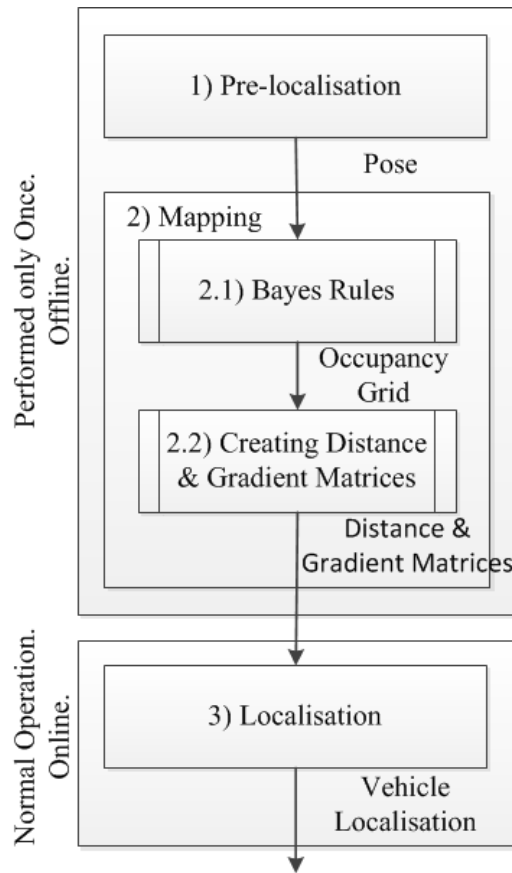


Fig. 3.1 Localization methodology described in the document.

As shown in Fig. 3.1, 1) *Pre-localization* aims to pinpoint the vehicle position, in a first stage, allowing the 2) *Mapping* procedure. In the mapping procedure, 2.1) *Bayes Rules* are applied in order to build the occupancy grid of the indoor environment.

After obtaining the occupancy grid, the distance transform and the Sobel filters are applied to the previously obtained 3D map (occupancy grid). That way, 2.2) *Creating Distance & Gradient matrices* block computes the distance and the gradient matrices, which are necessary during the normal vehicle operation, on-line, 3) *Localization*.

The procedure 1) *Pre-localization*, is explained in Chapter 7. The procedure and sub-procedures 2) *Mapping*, 2.1) *Bayes Rules* and 2.2) *Creating Distance & Gradient matrices* are

described in the Chapter and sub-sections 8, 8.1 and 8.2, respectively. Finally, the 3) *Localization* is presented, in detail, in Chapter 9.

### 3.1. Pre-Localization and Mapping Modules

Lets imagine a situation where it is possible to have a controlled scenario and, during a certain time, there are no dynamic objects or individuals crossing the environment. During this time, called by author as *preparation time*, the scenario is static, allowing to move the vehicle, across rooms, corridors and halls, and acquiring data without any disturbance being introduced by dynamic objects. The data includes odometry and *observation module* readings.

The offline SLAM procedure, represented in the Fig. 3.1 with the *pre-localization* block, can be applied using the data acquired during the *preparation time*. Once the SLAM has been applied, the 2D estimated map is used to determine the location of the vehicle. In summary, the application of the *pre-localization* procedure, works as an indoor "GPS", allowing to compute the vehicle pose.

Finally, still offline and with the position of the vehicle obtained, the three-dimensional map can be built and the respective distance and gradient matrices are stored. This is represented in the Fig. 3.1 with the *mapping* block.

These two procedures (*pre-localization* and *mapping*) will only be performed once. The stored distance and gradient matrices are used as look-up tables for the *localization* procedure, in the normal vehicle operation.

The movement performed during the *preparation time* is not autonomous; it must be performed through the use of a remote control. The user that controls the vehicle should be able to navigate in zones of interest, which are the ones that have objects, walls and corners, to map. These zones are helpful in the robot's self-localization.

It was developed a rotating LRF (tilting LRF), called *observation module*, described in Chapter 5, which acquires three-dimensional points. Using this *observation module*, at each scan, it is possible to determine linear segments (walls, doors and furniture) [57] and points that are invariant with the vehicle displacement (corners and columns) [59]. The observation module, as will be explained with more detail in Chapter 5, comprises a servo DC motor, which enables the LRF rotation, acquiring points with coordinates  $(x, y, z)$ .

In fact, the *pre-localization* phase, the SLAM procedure has two different steps:

1) Initially, with the observation module fixed in the horizontal (tilting LRF in the horizontal), and controlling the vehicle with a joystick, the acquired data in the two-dimensional space can be logged. The acquired points are reduced into linear segments and points allowing the application of the EKF-SLAM step to build a 2D feature map.

2) After, with the *observation module* rotating, also controlling the vehicle with a joystick, 3D data can be logged to be applied with an EKF-Loc procedure, which only estimates the vehicle state, with the prior knowledge about the 2D feature map obtained in the EKF-SLAM step. The EKF-Loc procedure allows to locate accurately the vehicle pose, becoming possible the 3D mapping.

In fact, it is necessary to create the feature map firstly, with an EKF-SLAM algorithm, without any prior knowledge about the surrounding environment. To do that, it is preferable, for a more accurate result, to use the maximum possible data, about the environment. This maximum possible data is obtained with the tilting LRF fixed on the horizontal.

Only after, with the prior knowledge about the 2D feature map, the vehicle can be accurately located and perform the 3D mapping. To do that, a simpler EKF (called EKF-Loc) is used, and the tilting LRF is in the rotating mode. Therefore, the acquired points need to be projected in the ground plan, allowing to apply the EKF-Loc step with linear segments and invariant points in the two-dimensional space, while the same points in the three-dimensional space allows the 3D mapping.

Lets consider now the following assumption is assumed: in an indoor environment, when the scenario is controlled, i.e. without dynamic objects crossing the environment, the scenario is similar between the  $z$  coordinates:  $Z_{min}$  and  $Z_{max}$  metres, such is example a corridor as shown in Fig. 3.2.

Lets consider the set of the points  $P$  acquired by the *observation module* with coordinates  $(x, y, z)$ . The projection in a 2D space of points  $P$  whose  $z$  coordinate is between  $Z_{min}$  and  $Z_{max}$  metres, constitute the set of data points  $P_{2D}$ , with coordinates  $(x, y)$ , see Fig. 3.3.

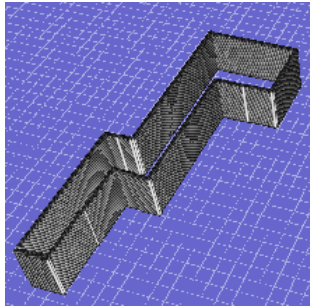


Fig. 3.2 Corridor, walls in the  $z$  coordinate gap of  $Z_{min} = 0.2$  metres to  $Z_{max} = 1.8$  metres.

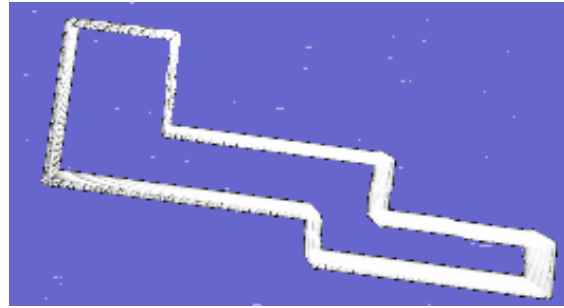


Fig. 3.3 Corridor (projection in a 2D space).

The data detection and extraction module, described in the Chapter 6, take the set of points  $P_{2D}$  and detect and extract information on features as walls, doors, furniture, corners and columns, which are used in the *pre-localization* procedure. The set of linear segments and invariant points found in  $P_{2D}$  are called  $Lin_{2D}$  and  $Pnt_{2D}$ , respectively.

In summary, the sets  $Lin_{2D}$  and  $Pnt_{2D}$  make it possible to pinpoint the vehicle position using the *pre-localization* and mapping the surrounding environment in the 3D space, at the same time, with the set of points  $P$ .

Eight different scenarios were mapped in the three-dimensional space using *pre-localization and mapping* procedures. The results are presented in Chapter 10.

### 3.2. Localization Module

The author belongs to the 5DPO team which participates in the Robotic Soccer Middle Size League (MSL) at the national and international RoboCup.

In the MSL football games, most of the teams have robots with Omni-directional camera sensors which provide an image of the football field, see Fig. 3.4 left. A segmented image is obtained from the image processing and it is possible to separate the image in colours and associate the colours to objects, see Fig. 3.4 right: ball, obstacles, white lines and the green field.

In the segmented image it is also possible to detect the transition between green-white and white-green, along the  $360^\circ$  camera field of view. Following a good calibration, these



transitions are transformed into point coordinates (2D points) of the field's white lines. These points are helpful for robot localization.

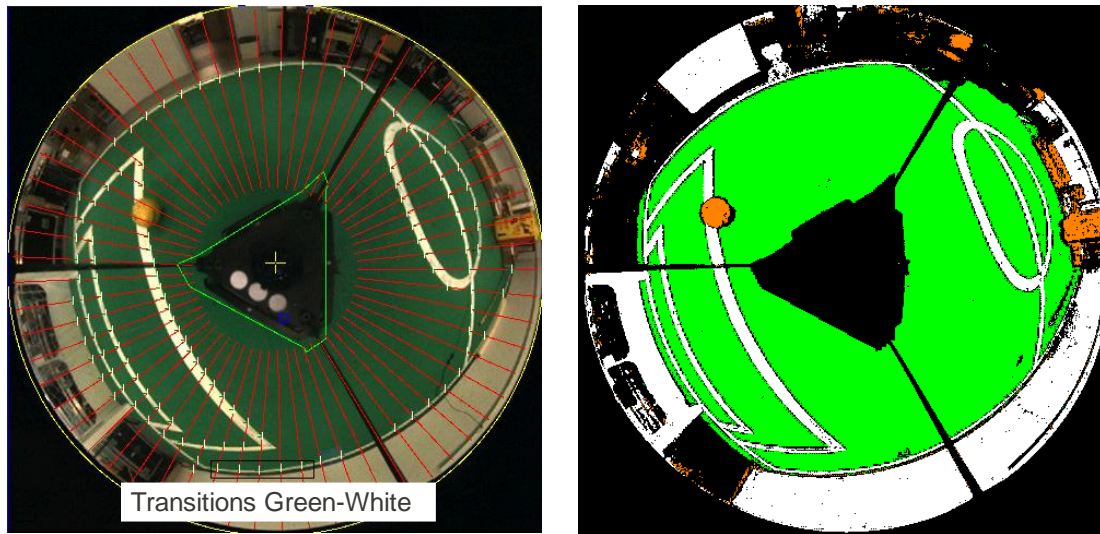


Fig. 3.4 Vision with a field of view of 360° (left), with the detection of green to white transitions. Segmented image on the right: green field (green), field lines (white), obstacles (black) and the ball (orange with circular form).

The localization algorithm used in the robotic football league, by the robots of the 5DPO team, fuses with an Extended Kalman Filter, the odometry data and the result of the Perfect Match algorithm, described by M. Lauer *et al.* [4].

The Perfect Match algorithm is a time saver matching algorithm, which is robust and feasible, and has proven its value in the Robotic Soccer Middle Size League in RoboCup.

Lets imagine now that the points of transitions are substituted by two-dimensional points obtained using an horizontal fixed Laser Range Finder. Furthermore, the field lines are substituted by LRF data obtained in the 2D plan that crosses the environment in the horizontal. Therefore, the same algorithm applied to the MSL, can be used to localize a mobile robot, equipped with a LRF, in indoor environments.

But, the environment must be static for the algorithm to operate in a more robust way. Therefore, there is a problem with this approach: when using the LRF, all the obstacles appear (static or dynamic). The dynamic obstacles introduce a lot of perturbation in the algorithm. In this case, using data obtained with the horizontal fixed LRF, the algorithm will not work properly, since the environment is not static.

However, the upper side of a room or corridor, including vertical walls and the ceiling above the normal height of a person (the headroom), can be considered almost a *static scenario*. This occurs because this area remains unchanged over long periods of time, i.e., objects are not frequently switched from one place to another. On the upper side of buildings - the *static scenario* - we can find walls, holes, ventilation or other types of air conditioned pipelines, windows, fire sensors and light supports. All of these "objects" are static infrastructure and helpful in the vehicle localization when a three-dimensional *observation module* is used.

If it was constructed previously, by the *pre-localization and mapping* procedure, a three-dimensional map of the *static scenario*, a 3D matching algorithm can be used to pinpoint the vehicle location, using the rotating Laser Range Finder data.

The *localization* using the 3D matching algorithm is described in detail in Chapter 9. Results on the application of the *localization* procedure in eight different scenarios, are presented in Chapter 10.

## 4. Simple Landmark-Based Localization

The *simple landmark-based* approach, applied to a Lego NXT robot (called LegoFeup), in a constrained and controlled scenario, is a first and simpler algorithm of localization, based on natural landmarks (smooth walls) that can be applied to some real environments.

The Lego NXT kit has three servomotors and sensors for touch, sound, ultrasound and light. Furthermore, the kit has a fundamental component: the NXT brick that enables the development of the robot's software by using, for instance, RoboLab or Lejos JAVA for Lego Mindstorms [60]. In complex and longer programming tasks, such as in the implementation of a Kalman Filter, an object-oriented language such as JAVA for Lego Mindstorms is preferred.

The LegoFeup, Fig. 4.1, is a differential vehicle, with two traction wheels and one free wheel. Each servomotor has an encoder capable of measure the number of turns of the corresponding wheel. It has two infrared sensors one at each extreme of the vehicle pointing for the side (IR Sharp [15]), which are able to measure distances relative to objects in the sensor's range (as walls). To implement this localization algorithm, the encoder's data together with the information of IR Sharp sensors was used.

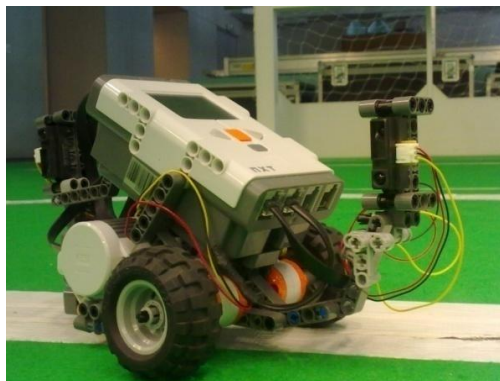


Fig. 4.1 LegoFeup. Lego NXT, brick, two IR SHARP sensors, two wheels (differential drive robot).

If the LegoFeup navigates inside a particular area, describing a random or a predefined trajectory (for instance a square, rectangle or circle), when predicting its position based on odometry, using encoders on each wheel, the error of the estimative of the position of the vehicle will increase without limits.

The LegoFeup has the ability to measure its distance and orientation with respect to landmarks (walls) every time that the landmark comes within the range of the infrared sensors. If there are white walls at known positions inside the area where the robot is navigating, its position can be corrected by implementing the Extended Kalman Filter (EKF)

as a probabilistic method. Therefore, in the routine of localization the vehicle position provided by odometry is corrected, using an EKF, every time a wall is seen.

A particular scenario was used during the work described here (see Fig. 4.2). Four walls are at known positions forming a square. The  $wall_{x+}$  and  $wall_{x-}$  are parallel to the x axis, with  $L_w$  and  $-L_w$  x coordinates, respectively, while the  $wall_{y+}$  and  $wall_{y-}$  are parallel to the y axis, with  $L_w$  and  $-L_w$  y coordinates, respectively, as a square room. The trajectory that the LegoFeup should follow is a square-shaped path of length  $L_p$ . In this particular case, the white walls are at the first metre of each square side.

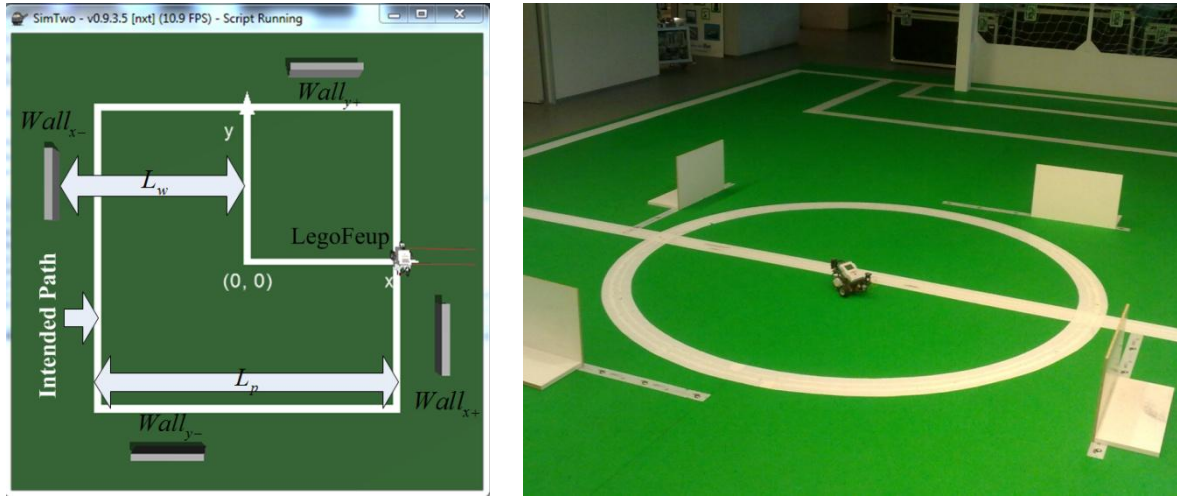


Fig. 4.2 Scenario. Left: simulation scenario. Right: real scenario.

The experiment was performed in a simulation scenario and in a real scenario (using the Lego NXT). The simulator SimTwo [61] used in the experiment takes into account collisions between rigid bodies and considers additive noise in sensor measurements, thus making it very realistic. The test in the simulation scenario is very important as a first validation. Although that, the use of a simulation scenario allows the test of algorithms, without the need of a real robot.

Therefore, the *simple landmark-based* approach was first tested in the simulation scenario, Fig. 4.2 left, as first validation stage. After, in a transparent way, the same algorithm of localization was tested in a real scenario, Fig. 4.2 right, with successful results.

#### 4.1. Algorithm Implementation

The algorithm of localization, called LegoFeup loop can be divided into four different modules: I) the vehicle path "Control Module", II) the "Feature Association Module", III) the "Observation Module", provided by the IR sensors and finally, IV) the "Estimation Module", using the Extended Kalman Filter (EKF). The entire LegoFeup loop is shown in Fig. 4.3.

The loop inputs are the value of the IR sensors and the number of pulses between two consecutive time steps,  $Odo_1$  and  $Odo_2$ . The sensor values are converted into distances,  $ds_1$  and  $ds_2$ , using the characteristic curve of the IR Sharp sensors.

If the distance value in both sensors is less than 30 centimetres it is considered that an observation was made. In that case, the observation is assigned to the correct feature (wall)

using the actual estimated state. Therefore, the observation  $Z_v(d_w, ObsAng)$  is generated by the "Observation Module" and passed to the "Estimation Module".

If there are no an observation, the "Estimation Module" only performs the prediction step. When an observation about a wall is available, the update step is also performed after the prediction step, using for that the observation  $Z_v$  generated by the "Observation Module".

Finally, the estimated vehicle state, will be used in the "Control Module", which deliver to the robot, the intended velocities for each wheel, aiming that it follows the intended path.

This approach has the fundamental goal of estimate the vehicle pose, the 2D coordinates  $(x_v, y_v)$  and the orientation relative to the x axis  $(\theta_v)$ . Along this chapter, the vehicle true pose is represented with the state  $X_v$  and state variables  $x_v, y_v$  and  $\theta_v$ . The estimated pose will be represented as  $\hat{X}_v$ , with estimated variables  $\hat{x}_v, \hat{y}_v$  and  $\hat{\theta}_v$ .

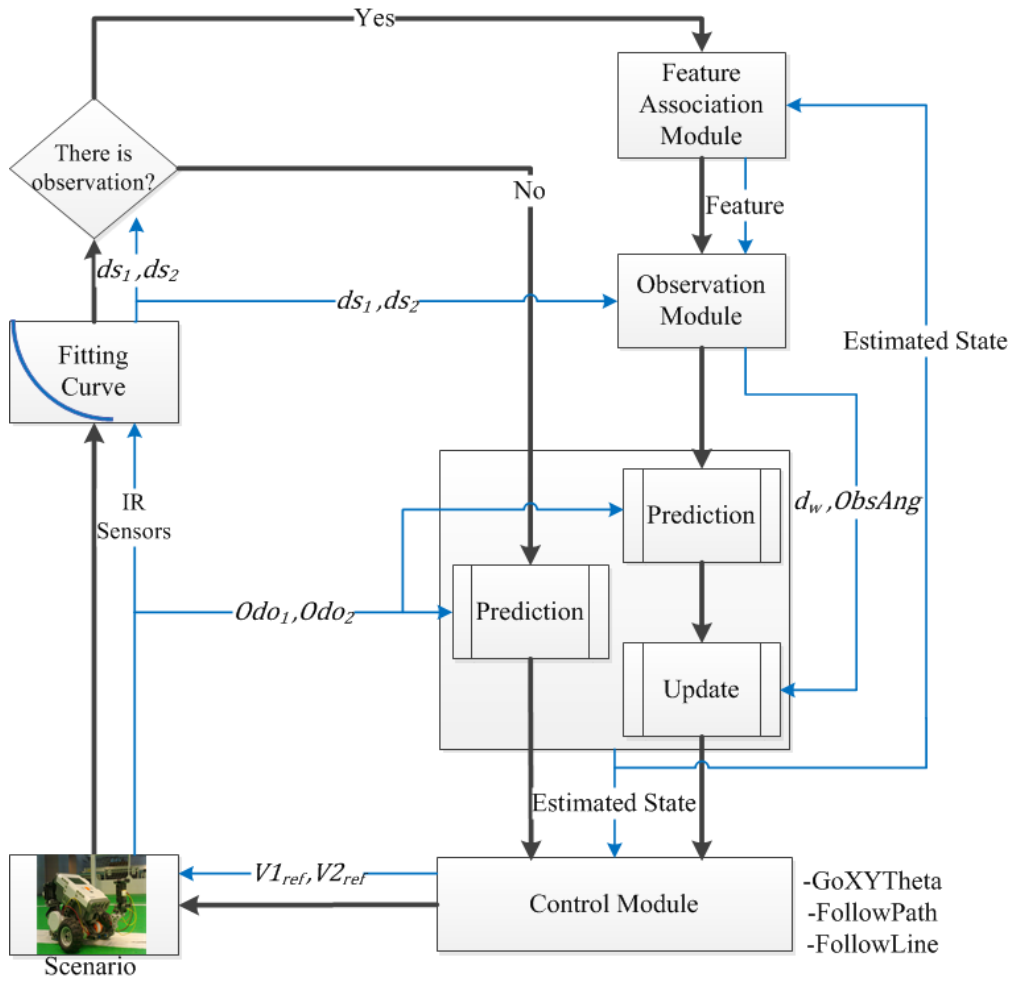


Fig. 4.3 LegoFeup loop. The blue line represents the data flow. The black and bold line shows the process flow.

#### 4.1.1. Control Module

The control module has the estimated state variables  $\hat{x}_v, \hat{y}_v$  and  $\hat{\theta}_v$  as inputs provided by the estimation module. The purpose of the module is to make the vehicle follow an intended path using the knowledge on the estimated state variables and a set of predefined routines: 1) "GoXYTheta", 2) "FollowLine".

The "GoXYTheta" routine allows the vehicle to reach a specific point in space, located at position XY, finishing with an orientation of Theta. Therefore, other routine inputs are the intended point XY and the desired final orientation.

The aim of the "FollowLine" is that the Lego NXT follows a line trajectory. In the case of the "FollowLine" routine, the other inputs are the line that is intended to be followed and the direction of the line (two points on the line).

Finally, the control module implements a "FollowPath" routine, where the basic routines described above are used to perform a path. An example of a path is shown in the Fig. 4.2 left. This path has a square shape and its sides are lines in the positive and negative directions  $x$  and  $y$ . All of these routines have the linear and rotation intended velocities ( $v_{ref}$  and  $\omega_{ref}$ ) as outputs, which are afterwards converted into intended velocities for each wheel using the following expressions:

$$V1_{ref} = v_{ref} + \frac{b \cdot \omega_{ref}}{2}, \quad V2_{ref} = v_{ref} - \frac{b \cdot \omega_{ref}}{2} \quad (4.1)$$

where  $b$  is the distance between wheels and  $V1_{ref}$  and  $V2_{ref}$  are the intended velocities for each wheel.

#### 4.1.2. Association Module

If the LegoFeup passes close enough to a wall that the infrared sensors are able to measure the distances, a new observation can be generated. Before generate the observation, it is necessary to assign the observation to the correct wall. In the case of the experimental work described in this chapter, the feature association module can be summarised by the diagram shown in Fig. 4.4.

If the vehicle detected a wall when its estimative of the state is equal to an  $\hat{x}_v$  and  $\hat{y}_v$  positives, the corresponding found wall is  $wall_{y+}$ . On the contrary, when the wall detection occurred when the vehicle estimated position is equal to  $\hat{x}_v$  negative and  $\hat{y}_v$  positive, the respective detected wall is the  $wall_{x-}$ . In the other side, if a observation of a wall occurred with the vehicle estimated position equal to  $\hat{x}_v$  and  $\hat{y}_v$  negatives, the corresponding detected wall is the  $wall_{y-}$ . Finally, the  $wall_{x+}$  can only be assigned to the new observation when, the vehicle estimated variables are:  $\hat{x}_v$  positive and  $\hat{y}_v$  negative.

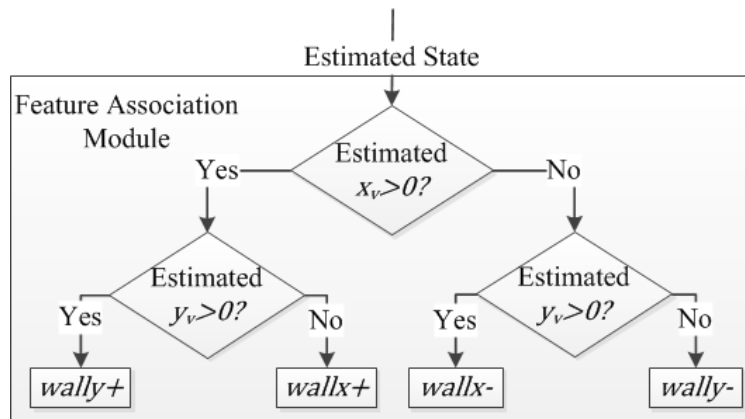


Fig. 4.4 Feature association module.

### 4.1.3. Observation Module

The IR sensor used in this experiment, Sharp IR Range Finder, measures distances between 4 and 30 centimetres. The characteristic curve of the Sharp IR sensors is represented by the graph shown in Fig. 4.5, obtained using experimental acquired data. It can be seen that the distance value (in centimetres) is non-linearly dependent on the sensor's analogue value. A candidate function that fits the sensor curve is given by the following expression:

$$D = a \cdot \left( \frac{C_M}{A - C_R} - b \right) \quad (4.2)$$

where  $A$  is the NXT read value and  $D$  is the distance.  $C_M$  is the constant of multiplication and  $C_R$  is the constant of linearisation. The constants  $a$  and  $b$  are the linear function constants.

In this figure (Fig. 4.5), it is shown the real acquired values and the results of the best relation that makes it possible to compute the distance using the analogue value of the Sharp. The best relation was obtained by optimizing, on *Matlab*, the sum of the quadratic error between the real acquired values and the fitting function shown at equation (4.2).

The corresponding parameters of the curve are shown in the following table, Table 4.1:

$a$	33.5409
$b$	0.0581
$C_M$	102
$C_R$	107

Table 4.1 Parameters of the fitting curve, shown in equation (4.2), corresponding to the sensor Sharp IR Range Finder.

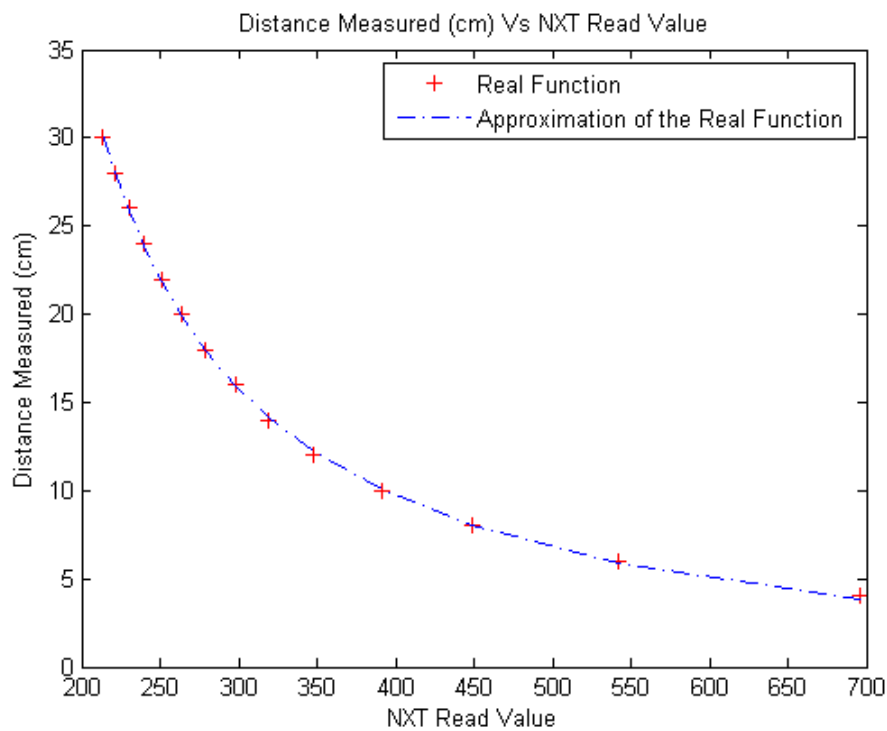


Fig. 4.5 Real and fitted curve.



Using two IR sensors, it is possible to measure the distance  $d_w$  and the orientation  $ObsAng$  of the robot related to the feature wall, represented at Fig. 4.6. These measures are used to correct the vehicle position, since the true location of the wall is known à priori. The observation is given by the following equations:

$$ObsAng = \tan^{-1} \left( \frac{ds_1 - ds_2}{L} \right) \quad (4.3)$$

$$d_w = \frac{ds_1 + ds_2}{2} \cdot \cos(ObsAng) \quad (4.4)$$

where  $ds_1$ , and  $ds_2$  are the inputs, measurements obtained with the IR sensors and  $L$  is the distance between the sensors, see Fig. 4.6.

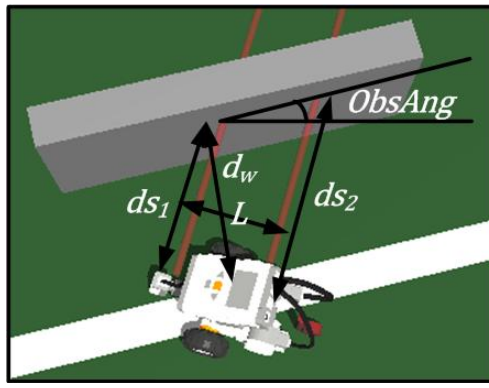


Fig. 4.6 Angle observed ( $ObsAng$ ) and distance to the wall ( $d_w$ ).

The error associated to the measurements  $ds_1$ , and  $ds_2$ , given by  $\varepsilon_{ds}$ , is modeled as Gaussian noise (with zero mean and covariance  $R_{ds}$ ). The observation noise vector is equal to:

$$\varepsilon_{ds} = [\varepsilon_{ds1} \quad \varepsilon_{ds2}]^T \quad (4.5)$$

An experiment was conducted aiming to characterize the variance of the IR Sharp. The obtained results helped to realize that the sensors variance has a little and negligible dependence with the variation of the measured distance.

The experiment consisted in, successively place a wall at a distance between 0.04 and 0.3 metres in intervals of 0.02 metres, in relation to an fixed IR sharp sensor. To each distance, was taken three samples of 1000 measurements.

The average of the quadratic difference between the true and the measured distances, in the three samples, for all the measurements acquired in the interval of 0.04 to 0.3 metres, is the variance considered for the IR Sharp sensor. The square root of the variance is equal to the standard deviation  $\sigma_{\varepsilon_{ds}}$ .

The experimental value obtained for  $\sigma_{\varepsilon_{ds}}$  is 0.001 metres. Therefore,  $R_{ds}$  can be written as a constant matrix, equal to:

$$R_{ds} = \begin{bmatrix} 0.001^2 & 0 \\ 0 & 0.001^2 \end{bmatrix} \quad (4.6)$$



Due to the maximum range of the infrared sensors, every time that the distance measured in both the IR sensors is less than or equal to 30 centimetres, it is considered that an observation has been made. The observation model  $h_v$  is given by the following equations, depending on what wall the observation is corresponding:

$$h_v(wall_{x+}) = h_v(wall_{x-}) = \begin{bmatrix} x_v \\ \theta_v \end{bmatrix} \quad (4.7)$$

$$h_v(wall_{y+}) = h_v(wall_{y-}) = \begin{bmatrix} y_v \\ \theta_v \end{bmatrix} \quad (4.8)$$

Therefore, the estimated observation module is obtained using the estimated state variables and can be written as follows:

$$\hat{h}_v(wall_{x+}) = \hat{h}_v(wall_{x-}) = \begin{bmatrix} \hat{x}_v \\ \hat{\theta}_v \end{bmatrix} \quad (4.9)$$

$$\hat{h}_v(wall_{y+}) = \hat{h}_v(wall_{y-}) = \begin{bmatrix} \hat{y}_v \\ \hat{\theta}_v \end{bmatrix} \quad (4.10)$$

The observation,  $Z_v$  can be written as the following equation, as function of the observation module:

$$Z_v = h_v + r \quad (4.11)$$

where  $r$  is the observation error, modelled as gaussian noise with zero mean and covariance equal to  $R$ .

At each wall only two state variables can be observed. Therefore, the observation  $Z_v$ , can be obtained with the use of the following equations:

$$Z_v(wall_{x+}) = \begin{bmatrix} L_w - d_w \\ \frac{\pi}{2} - ObsAng \end{bmatrix} \quad (4.12)$$

$$Z_v(wall_{x-}) = \begin{bmatrix} -(L_w - d_w) \\ -\frac{\pi}{2} + ObsAng \end{bmatrix} \quad (4.13)$$

$$Z_v(wall_{y+}) = \begin{bmatrix} L_w - d_w \\ \pi + ObsAng \end{bmatrix} \quad (4.14)$$

$$Z_v(wall_{y-}) = \begin{bmatrix} -(L_w - d_w) \\ ObsAng \end{bmatrix} \quad (4.15)$$

Therefore, using the observation vector  $Z_v$ , the matrix  $R$  can be computed. The matrix  $R$  is given at each time step with the following equation:

$$R = \frac{\partial Z_v}{\partial \varepsilon_{d_s}} R_{d_s} \frac{\partial Z_v^T}{\partial \varepsilon_{d_s}} \quad (4.16)$$

where the gradient of the observation in order to the measurements ( $ds_1$  and  $ds_2$ ) noise ( $\varepsilon_{d_s}$ ), is given by the following equations:

$$\begin{aligned} \frac{\partial Z_v}{\partial \varepsilon_{d_s}} &= \begin{bmatrix} \frac{\partial Z_v^1}{\partial d_{s1}} & \frac{\partial Z_v^1}{\partial d_{s2}} \\ \frac{\partial Z_v^2}{\partial d_{s1}} & \frac{\partial Z_v^2}{\partial d_{s2}} \end{bmatrix} = \begin{bmatrix} \text{signal} \cdot \frac{\partial d_w}{\partial d_{s1}} & \text{signal} \cdot \frac{\partial d_w}{\partial d_{s2}} \\ \frac{\partial ObsAng}{\partial d_{s1}} & -\frac{\partial ObsAng}{\partial d_{s1}} \end{bmatrix} \Leftrightarrow \\ \frac{\partial Z_v}{\partial \varepsilon_{d_s}} &= \begin{bmatrix} \frac{\text{signal} \cdot (e_1 + e_2)}{L} & \frac{\text{signal} \cdot (e_1 - e_2)}{L} \\ \frac{L}{L^2 + (d_{s1} - d_{s2})^2} & -\frac{L}{L^2 + (d_{s1} - d_{s2})^2} \end{bmatrix} \end{aligned} \quad (4.17)$$

where  $Z_v^1$  and  $Z_v^2$  are, respectively, the first and second elements of the vector  $Z_v$ . The variable *signal* is equal to  $-1$  when the observed walls are  $wall_{x+}$  or  $wall_{y+}$ . On the contrary, when the observed walls are  $wall_{x-}$  or  $wall_{y-}$ , *signal* is equal to  $1$ . The expressions of  $e_1$  and  $e_2$  are equal to the following equations:

$$e_1 = \frac{1}{2} \cos(ObsAng) \quad (4.18)$$

$$e_2 = -\frac{d_{s1} + d_{s2}}{2} \cdot \sin(ObsAng) \cdot \frac{L}{L^2 + (d_{s1} - d_{s2})^2} \quad (4.19)$$

#### 4.1.4. Estimation Module

The Extended Kalman Filter (EKF), as shown in Algorithm 4.1, consists of two steps that work in a cycle. First, the vehicle state and covariance is predicted (EKF prediction). Then, if there is a new observation, this is assigned to the correct feature and the vehicle state and covariance are corrected (EKF update).

Algorithm 4.1 Extended Kalman Filter.

LegoFeupCycle
$\hat{X}_v(0), P_v(0) \leftarrow (X_{initial}, P_{initial})$ <b>loop</b> <b>EKF Prediction</b> $\hat{X}_v(k+1 k), P_v(k+1 k) \leftarrow \text{Predict}(Odo, \hat{X}_v(k k), P_v(k k))$ <b>New Observations?</b> $[Z_v] \leftarrow \text{Measures}$ $[\hat{h}_v] \leftarrow \text{Observation model}(\hat{X}_v(k+1 k))$ <b>EKF Update</b> $\hat{X}_v(k+1 k+1), P_v(k+1 k+1) \leftarrow \text{Update}(Z_v, \hat{h}_v, \hat{X}_v(k+1 k), P_v(k+1 k))$ <b>end loop</b>

The EKF is a Kalman Filter estimator in which the non-linear kinematic and observation models are transformed into linear models using the Taylor expansion to update the covariance (see chapter "Nonlinear Estimation" in [53]).

The prediction step uses the kinematic model of the vehicle with odometry data to estimate, at each time step, the vehicle's new state. In this stage, it is computed the new estimative of the state and covariance matrix,  $\hat{X}_v(k+1|k)$  and  $P_v(k+1|k)$ , respectively, using for that the previous  $\hat{X}_v(k|k)$  and  $P_v(k|k)$ .

In discrete time, the dynamics of the vehicle, can be described by its kinematic model that is based on the transition function. Therefore, the vehicle new state comes equal to:

$$X_v(k+1|k) = f_v(X_v(k|k), Odo, q_v) \quad (4.20)$$

where the transition function, based on the centred differences, is given by:

$$f_v(X_v(k|k), Odo, q_v) = \begin{bmatrix} x_v \\ y_v \\ \theta_v \end{bmatrix}(k|k) + \begin{bmatrix} d \cdot \cos\left(\theta_v(k|k) + \frac{\Delta\theta}{2}\right) + \varepsilon_{xv} \\ d \cdot \sin\left(\theta_v(k|k) + \frac{\Delta\theta}{2}\right) + \varepsilon_{yv} \\ \Delta\theta + \varepsilon_{\theta v} \end{bmatrix} \quad (4.21)$$

where the index  $(k+1|k)$  indicates the present time step, while  $(k|k)$  represents the previous time step. The term  $\Delta\theta/2$  is introduced here, aiming to decrease the error introduced by the discretization, [62].

As shown in the following figure, Fig. 4.7, the vehicle displacement between consecutive time steps, as it moves forward is represented by  $d$  and the rotation by  $\Delta\theta$ . These variables are computed using the encoder data as described in the following Chapter 5, for the differential drive robot, the RobVigil. These variables can be represented with the following vector:

$$Odo = [d \quad \Delta\theta]^T \quad (4.22)$$

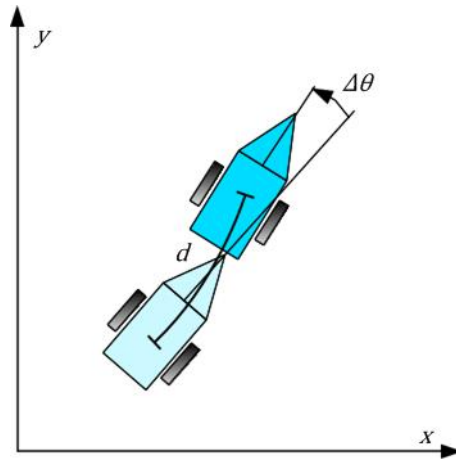


Fig. 4.7 Vehicle kinematics. Forward distance and rotation travelled,  $d$  and  $\Delta\theta$ , between consecutive cycle times.

The error in the odometry ( $q_v$ ), is modelled as Gaussian noise (with mean zero and covariance matrix equal to  $Q_v$ ). The error  $q_v$  is represented by the vector:

$$q_v = [\varepsilon_{xv} \quad \varepsilon_{yv} \quad \varepsilon_{\theta v}]^T \quad (4.23)$$

The estimated state after the prediction step is equal to the estimated transition function:

$$\hat{X}_v(k+1|k) = f_v(\hat{X}_v(k|k), Odo, \hat{q}_v) \quad (4.24)$$

As the error in the odometry ( $q_v$ ), is modelled as Gaussian noise, i.e. with zero mean, the vector  $\hat{q}_v$  is only constituted by zeros. Therefore, the new estimative of the vehicle state after the prediction step comes equal to:

$$\hat{X}_v(k+1|k) = \begin{bmatrix} \hat{x}_v \\ \hat{y}_v \\ \hat{\theta}_v \end{bmatrix}(k|k) + \begin{bmatrix} d \cdot \cos\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \\ d \cdot \sin\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \\ \Delta\theta \end{bmatrix} \quad (4.25)$$

During the prediction step, the covariance matrix  $P_v(k+1|k)$  is computed using the following expression:

$$P_v(k+1|k) = \frac{\partial f_v}{\partial X_v} P_v(k|k) \frac{\partial f_v^T}{\partial X_v} + \frac{\partial f_v}{\partial q_v} Q_v \frac{\partial f_v^T}{\partial q_v} \quad (4.26)$$

where the gradient of the vehicle's kinematics model in order to the noise on the odometry input ( $\frac{\partial f_v}{\partial q_v}$ ), is equal to the identity matrix. The gradient of the vehicle's kinematic model in order to the estimated state is equal to the following matrix:

$$\frac{\partial f_v}{\partial X_v} = \begin{bmatrix} 1 & 0 & -\frac{d}{2} \cdot \sin\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \\ 0 & 1 & +\frac{d}{2} \cdot \cos\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \\ 0 & 0 & 1 \end{bmatrix}_v \quad (4.27)$$

The prediction step, i.e. computation of  $\hat{X}_v(k+1|k)$ ,  $Q_v$  and  $P_v(k+1|k)$ , applied to differential robots is explained with more detail in the Chapters that describes the *pre-localization* and the *localization* (Chapter 7 and 9).

The state estimative ( $\hat{X}_v$ ) update is carried out during every cycle when an observation is obtained, after applied the prediction step. The estimative of the state, after the update step is equal to the following equation:

$$\hat{X}_v(k+1|k+1) = \hat{X}_v(k+1|k) + W(k+1) \cdot V(k+1) \quad (4.28)$$

where  $W(k+1)$  is the Kalman Gain and  $V(k+1)$  the innovation. The innovation, depending on the observed wall (see equations (4.9) and (4.10) for  $\hat{h}_v$ , and see equations (4.12) to (4.15) for  $Z_v$ ), is given by the equation:

$$V(k+1) = Z_v - \hat{h}_v \quad (4.29)$$

The Extended Kalman Filter Gain is equal to the following expression (see Chapter "Nonlinear Estimation" in [53]):

$$W(k+1) = P_v(k+1|k) \frac{\partial h_v^T}{\partial X_v} \left[ \frac{\partial h_v}{\partial X_v} P_v(k+1|k) \frac{\partial h_v^T}{\partial X_v} + R \right]^{-1} \quad (4.30)$$

where  $P_v(k+1|k)$  the covariance matrix computed during the prediction step. The gradient of the observation model  $h_v$  in order to the state,  $\frac{\partial h_v}{\partial X_v}$ , is equal to:

$$\frac{\partial h_v}{\partial X_v} = \begin{bmatrix} \frac{\partial h_v^1}{\partial x_v} & \frac{\partial h_v^1}{\partial y_v} & \frac{\partial h_v^1}{\partial \theta_v} \\ \frac{\partial h_v^2}{\partial x_v} & \frac{\partial h_v^2}{\partial y_v} & \frac{\partial h_v^2}{\partial \theta_v} \end{bmatrix} = \begin{bmatrix} C_1 & C_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.31)$$

where  $h_v^1$  and  $h_v^2$  are, respectively, the first and second elements of the vector  $h_v$ . When the walls that are seen are  $wall_{x+/-}$ , then  $C_1 = 1$ ,  $C_2 = 0$ . In the case of  $wall_{y+/-}$ , then  $C_1 = 0$ ,  $C_2 = 1$ . The matrix  $R$  is computed using the equation (4.16).

The covariance update is performed taking the following equation into account (see chapter "Nonlinear Estimation" in [53]):

$$P_v(k+1|k+1) = \left[ I - W(k+1) \cdot \frac{\partial h_v}{\partial X_v} \right] \cdot P_v(k+1|k) \quad (4.32)$$

where  $I$  the is the identity matrix with dimension  $3 \times 3$ .

## 4.2. Simulation and Real Scenario Communication

Both scenarios, real and simulation, are able to exchange information with a host computer on which the LegoFeup Loop algorithm is running. The information exchanged is: 1) the vehicle's sensor measurements that goes from the scenario to the LegoFeup Loop algorithm (the encoder values of each wheel and the infrared distances); 2) in the opposite direction, from the LegoFeup Loop to the scenario, the velocity of each wheel calculated by the Control Module algorithm is sent,  $V1_{ref}$  and  $V2_{ref}$ .

Fig. 4.8 shows the structure of communication between the EKF algorithm, the interface and the scenario. This figure also shows the transparent change between the real (Bluetooth) and simulation (UDP) scenarios.

The user interface, as shown in Fig. 4.9, Fig. 4.10 and Fig. 4.11, makes it possible to "start" and "stop" the algorithm. This interface also makes it possible to "set" the EKF configuration: 1) the covariance matrix  $Q_v$ ; 2) the sensor covariance  $R_{d_s}$ ; 3) the initial covariance; 4) the initial vehicle location; 5) the trajectory controller parameters, and finally 6) the vehicle reference velocities.

The interface was developed to show the Extended Kalman Filter variables: 1) the covariance matrix, 2) the covariance ellipse, 3) the vehicle's estimated position. The interface also shows the reference velocities at each vehicle's wheels  $V1_{ref}$  and  $V2_{ref}$  and the values of the infrared sensors,  $ds_1$  and  $ds_2$ . This interface communicates with the LegoFeup loop algorithm via UDP. The form, called "FEllipses", Fig. 4.11, shows the intended path, the vehicle's estimated position and finally, the vehicle's estimated orientation. It is also possible to see the covariance ellipse with a confidence of 95.4% (two times the standard deviation).

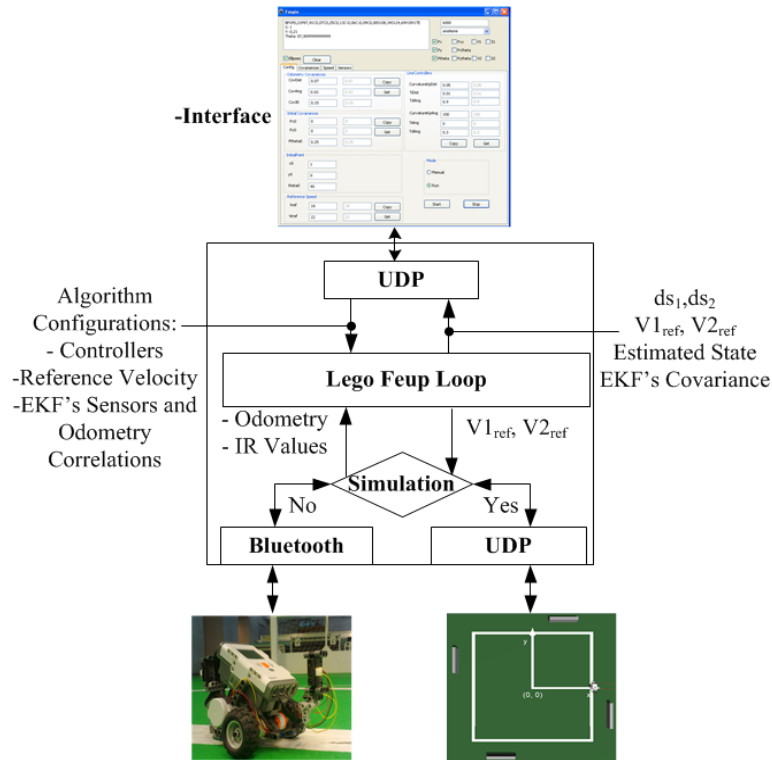


Fig. 4.8 Algorithm, communication and interface.

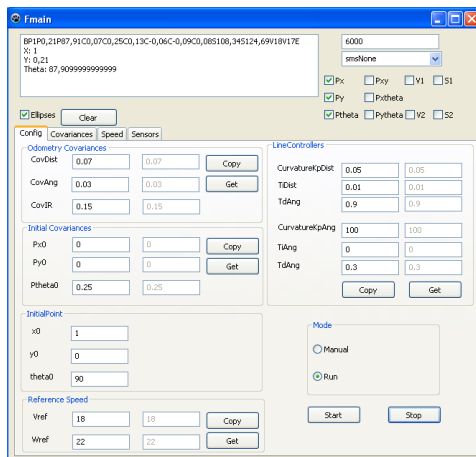


Fig. 4.9 Main Screen - Configuration Tab.

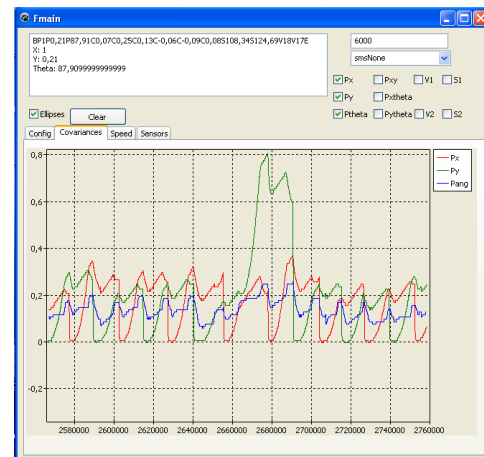
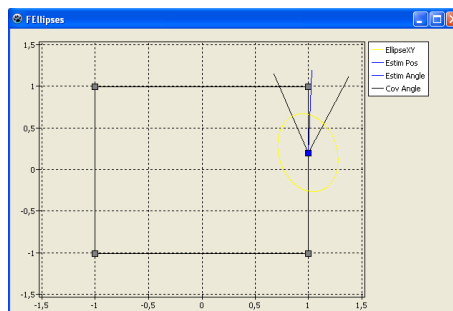
Fig. 4.10 Main Screen - here, the covariance in x, y and  $\theta$  directions are plotted.

Fig. 4.11 Main Screen - here, the vehicle position, angle and covariance ellipse is plotted.

### 4.3. Application on Education

Later, the *simple landmark-based* approach, applied to the simulation and real scenarios, was used to teach concepts of sensor fusion, navigation and probabilistic localization using EKF, in the course of Autonomous Robotic Systems, at the Electrical and Computer Engineering curriculum, in the Faculty of Engineering of the University of Porto.

There are other works describing the use of Lego Mindstorms in teaching programming, artificial intelligence and embedded systems to first-year engineering students [63] to [66], but none of them cover localization of a robot, based on an EKF.

Behrens *et al.* [63], believe that no matter what students are taught and in how much detail they are taught, they will only learn when the lectures come home to them, as they are put in the engineer's role.

Lego Mindstorms makes it possible to build embedded systems without any prerequisite knowledge, and to solve real problems with real constraints. S. H. Kim and J. W. Jeon [64], describe how freshmen engineers learn embedded systems using the Lego Mindstorms as a tool.

S. Sharad [64], describes the architecture of the Lego Mindstorm NXT embedded system and its effectiveness when used to teach embedded systems and engineering concepts.

In fact, the Lego Mindstorms are widely used in education and outreach activities as a way of teaching and simultaneously motivating students. However, they tend to be used only during the first years of the engineering curriculum. With the *simple landmark-based* approach, applied to teaching, is proposed an experiment to teach more complex topics, such as localization and estimation based on EKF, to older undergraduate students.

Other work related with this approach is described by J. Gonçalves's PhD thesis, chapter 6, [67], which uses IR Sharp sensors, in a small omnidirectional robot, aiming pinpoint its location using the walls of a maze.

The success of the experiment realized with the students, aiming their better understanding of Kalman Filters applied to mobile robots, was the core of the inspiration for the published work on the IEEE Transactions on Education [68].

The developed approach was described in this paper, as an implementation of a Multi Sensor Fusion System, which using an Extended Kalman Filter (EKF) and an association module, performs the fusion between the odometry and the obtained observations.

The experience of teaching undergraduate students was also described in this paper and it can also be used as a guide to undertake a pedagogical experiment involving the concepts of sensor fusion, navigation and probabilistic localization using an Extended Kalman Filter.

In order to help undergraduate students understand the EKF concepts more clearly, the user interface at Fig. 4.9, Fig. 4.10 and Fig. 4.11 was used, allowing students to tune the filter, and observe the immediate effect on the Extended Kalman Filter's behaviour. Therefore, this module provides students with the opportunity to learn about EKF operation, the importance of the covariance matrices, of the odometry and sensor noise, and how these affect the operation of the filter.

The topic "Localization of Mobile Robots using an Extended Kalman Filter in a Lego NXT" was taught in three-hour lectures. During the first lecture the theoretical content was

presented including the theory underlying the Kalman Filter, including its mathematical formulation.

The second and third lectures consisted of teaching the practical module, with students performing the experiment while working in pairs. During the first practical lecture, the students familiarised themselves with the LegoFeup tool and with the simulation scenario. After gaining an understanding of the algorithm/tool, an experiment in the simulation scenario was performed which was based on the presented guide. The third and last lecture finished the test by configuring the EKF as performed in the simulation scenario, but this time in the real scenario. The test was performed one group at a time.

In the simulation scenario, Gaussian noise with zero mean and standard deviation equal to  $\sigma_{\varepsilon_{ds}} = 0.001$  metres was introduced in the Sharp IR sensor readings. In the simulation it is possible to select turn on or off the Gaussian noise introduced in the Sharp IR sensors.

A Gaussian noise with zero mean and standard deviation of  $\sigma_d = 0.18$  metres for each metre travelled by the vehicle was introduced in the simulation scenario. Also a Gaussian noise with zero mean and standard deviation of  $\sigma_\theta = 0.029$  radians for each rotated radian was introduced. These errors simulate the error in the odometry that appears with the wheels' radius changes and caused by the variation of the distance between wheels. These values are measured experimentally and proved to be close to the real behaviour of the vehicle.

The experiment proposed to the undergraduate students consisted of the following steps; first in the simulation scenario: a) Executing a square trajectory based only on the odometry; b) Executing the same trajectory with direct localization, sensors' covariance matrix  $R_{ds}$  with very low values, based on the Sharp IR measurement without noise (turning "off" the noise); c) Adding Gaussian noise in the Sharp IR sensors' readings (turning "on" the noise) and repeating the previous step; d) Putting the EKF's sensors' covariance matrix  $R_{ds}$  equal to the Gaussian noise introduced in the simulation scenario. Giving the odometry covariance matrix  $Q_v$  a very low value and completing the same trajectory; e) Changing the value of the covariance matrix  $Q_v$ , to a very high value and repeating the previous step; f) Using the results of the previous steps to tune the filter to the best solution for  $Q_v$ . This last test and respective filter tuning should be kept and tested in the real scenario.

The students were given instructions on how to perform each step. Each group delivered a report at the end of the lecture explaining their vehicle's behaviour at each step, and justifying their conclusions.

The students should have concluded in step a) that the vehicle covariance should increase without bounds due to the absence of observations. The vehicle does not update or correct its position with the walls and, consequentially, it will be rapidly lost; in step b) that the user has a high level of confidence in the infrared sensor readings at the EKF filter, and the update is done directly with 100% confidence. At this stage the algorithm localizes the vehicle with success but only because the scenario is ideal, i.e. the Sharp IR measurements are obtained without noise; in c) that confidence in the infrared sensor readings at the EKF filter is high, and the update is done directly with 100% confidence. However, in this case the algorithm does not localize the vehicle correctly because the scenario in this step is not ideal, i.e., the infrared sensors have noise; in d) that the filter should not work since the sensors' measurements are rejected because confidence in the odometry is high. Then the localization



is performed based only on the odometry; in e) that the simulated situation is similar to that of step b). Now, instead of having high confidence in the sensors, the user has really low confidence in the odometry when compared with the infrared sensors. This situation leads the filter to update the vehicle localization directly, with a confidence near to 100% when a wall is observed.

Videos and theoretical content on the implementation of the experiment described here can be found in [2]. Furthermore, an application of the EKF algorithm can be downloaded with some tests that can be performed in the simulation scenario. The theory taught during the theoretical lecture is also available.

#### 4.4. Results on Education

The conduction of a student questionnaire indicated the impact that the use of the *simple landmark-based* approach had on the students' understanding, during the Autonomous Robotic Systems course. Most of the students had heard about Kalman Filters, but had never implemented them and did not understand how they operated or what their purpose was. By the end of the course, most students had completed the fundamental goal; they understood the Extended Kalman Filter and the localization based on the EKF as a probabilistic method of making the fusion between the odometry and the Sharp IR observations.

On the anonymous student survey, questions such as: "Are you now more interested in mobile robotics than before the lectures?", "Do you think that the experiment and the tutorial will be helpful for future studies on mobile robotics topic?" or finally, "Do you now understand, and feel capable of describing, the method of navigation and localization implemented?" received mostly positive answers.

The non-anonymous reports delivered by each group at the end of the practical lectures were evaluated, and lead to the conclusion that the experiment was a success. The undergraduate students understood theoretical concepts based on a practical tool. According to the experimental guide presented in the previous sub-section, the reports showed a percentage of satisfactory answers, in each step of the experiment, as shown in Table 4.2.

Answer	Percentage of correct answers
a)	90%
b)	70%
c)	80%
d)	80%
e)	80%

Table 4.2 Percentage of correct answers plotted.

At the end of the semester, the students sat a final exam for the course. The exam had three questions relating to the experiment and the Extended Kalman Filter applied to robot localization. By comparison with previous years' exams (where this tool was not used), the percentage of correct answers for this topic increased by about 20% with the introduction of the proposed tool in the Autonomous Robotic Systems lectures. Furthermore, analysing the correct answers highlights two aspects: consistency of reasoning and a correct visualisation of

the matter. These results prove that the method presented improves the students' understanding.

## 5. Hardware Setup

The robot platform used in the tests on the *three-dimensional map-based* approach, presented and developed during this PhD, was a differential wheeled traction vehicle, shown in Fig. 5.1, called RobVigil.

### 5.1. Robotic Platform

The main application of the RobVigil is the surveillance of public facilities as department stores, hospitals or others service scenarios. The use of a robot of this type, in these scenarios, allows systematic and uninterrupted inspection routines, with minimum human intervention.

When using a robot with these characteristics, the watcher is then free to perform the simpler and safety task of supervise alarms and the different cameras streams, sent by the robot during its normal operation.

Therefore, this vehicle has the fundamental goal of operating inside dynamic scenarios, with lots of people and dynamic objects crossing its navigation area. Then, it is convenient that the *localization* process only uses the headroom of a building (a static scenario), thus avoiding the influence of the disturbance introduced by dynamic objects.

The robot RobVigil is a differential wheel traction robot, with two traction wheels and a free wheel. The robot has a Laser Range Finder placed in a servo DC motor, allowing to acquire three-dimensional data on the surrounding environment. The developed tilting Laser Range Finder is used for mapping and localization and it is placed on the top of the vehicle.



Fig. 5.1. Left: the RobVigil robot equipped with the observation module. Middle: RobVigil robot in a surveillance task. Right: the robot without covering. The developed *observation module* can be seen on the top (vehicles head).

Its features include a low power processor onboard PC (Mini ITX, EPIA M10000G with a processor of 1.0GHz) and a height of approximately 1.3 metres, with the *observation module*

(tilting LRF) placed on the top. The LRF has a maximum range of 5 meters, as shown in the following figure, Fig. 5.2.

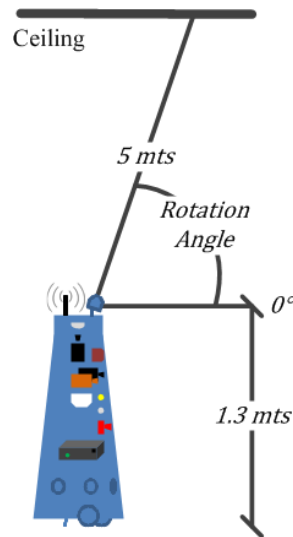


Fig. 5.2. RobVigil robot characteristics: 1.3 meters of height, the tilting LRF on the top with a range of 5 meters .

Furthermore, the RobVigil is equipped with sensors to detect hazardous situations, like fires, floods, gas leaks or movement. It is also equipped with three surveillance cameras, including an omnidirectional, an high resolution and finally, a thermal camera. This equipment is necessary to perform the surveillance task.

It also has ultrasound and infrared sensors around its body, allowing to navigate safely, while avoiding obstacles, during autonomous inspection routines. Besides, the robot has a communication interface and GUI interface, allowing the supervision of information as alarms and the different cameras in real time. The following figure, Fig. 5.3 shows the robot RobVigil onboard sensors and characteristics.

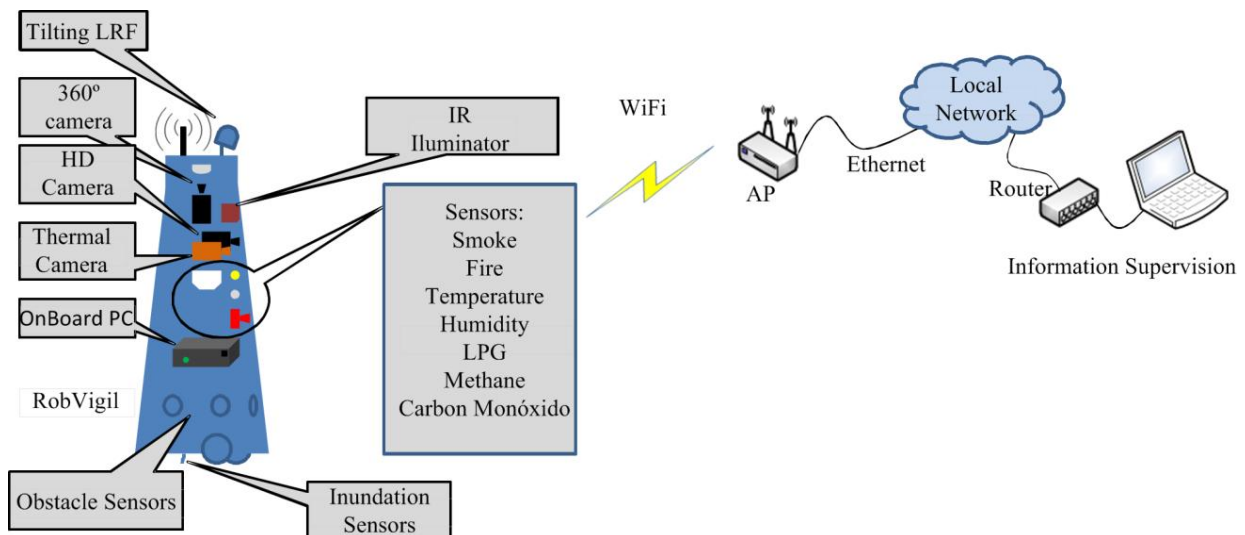


Fig. 5.3. RobVigil Robot. Its features and sensors.

## 5.2. Tilting Laser Range Finder

The 2D Laser Range Finder (LRF) – *Hokuyo URG-04LX-UG01* – was used to perceive the surrounding environment. To obtain a three-dimensional sensor, a tilt platform was created based on the high resolution dc servo motor, the *AX-12 Dynamixel Bioloid*. The complete LRF solution is shown in Fig. 5.4.

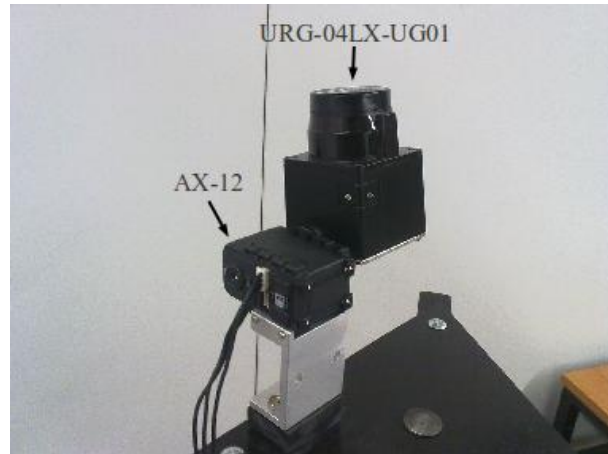


Fig. 5.4. The *observation module* developed.

The *AX-12* servo dc motor can rotate about  $270^\circ$  with a  $0.29^\circ$  of resolution. Its maximum speed is 114 rpm. The rotating platform created, made the LRF scanning in the horizontal possible, i.e. the tilt scanning. This allows the 2D laser to point downwards or backwards.

With a sample rate of 100 milliseconds, at each scan, the LRF gives 769 points. About these 769 points, the detection range begins at point 44 and ends at point 725. Therefore, among the acquired 769 points, only 682 are on the surrounding environment and then only these points can be used. As the LRF's angular resolution is of about  $0.3516^\circ$ , the corresponding range angle is of about  $239.79^\circ$ . The LRF rotation speed is fixed and equal to 60 rpm. The maximum number of points that can be obtained in a scan, where the *AX-12* rotates at a velocity of 1 rpm, is 344250 points and its duration is 45 seconds. Furthermore, when the servo speed is 114 rpm, the total scanning duration is of about 0.39 seconds and the number of points reached is 3984.

There is a transformation that takes the measurement of the LRF in polar coordinates and places it in the 3D space. The transformation is modelled as a set of homogeneous transformations (rotations and translations).

The following figure, Fig. 5.5, represents the laser (L) and the measurement (D) frames. In this figure, one reading of the point  $P$  in polar coordinates  $(d, \Psi)$ , represented with a blue point is also represented. The referential L is represented in green colour, while the referential D is represented in grey.

The referential L has origin at the centre of the inferior base, with cubic shape, of the Laser Range Finder, represented in Fig. 5.5. The referential D has origin at the laser beam source.

Therefore, the point  $P$  written in the coordinates of the referential D, represented as  $P^D$ , appears as a function of the laser's distance reading  $d$ , equal to:

$$P^D = [d \ 0 \ 0]^T \quad (5.1)$$

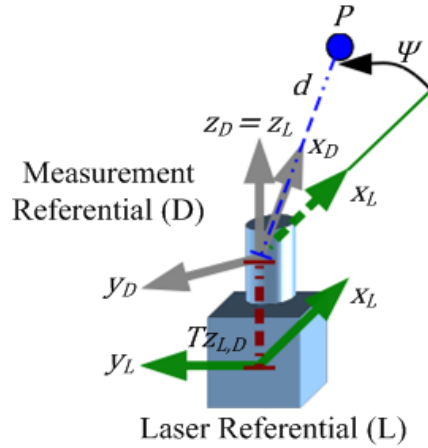


Fig. 5.5 Laser referential (L). Measurement Referential (D). Point P and reading acquired with the LRF ( $d$  and  $\Psi$ ).

The  $z$  axis of the referential L and D are coincident. The rotation of the yaw angle  $\Psi$ , between the referential D, in relation to the laser referential L is represented by the matrix  $R_D^L$ , and is expressed in the referential L:

$$R_D^L = \begin{bmatrix} \cos \Psi & -\sin \Psi & 0 \\ \sin \Psi & \cos \Psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.2)$$

The translation of the referential D, in relation to the referential L, also written in coordinates of the referential L, is expressed by the vector  $T_D^L$ , and is provided by the following expression:

$$T_D^L = [0 \ 0 \ T_{z_{L,D}}]^T \quad (5.3)$$

in which the distance  $T_{z_{L,D}}$  is represented in Fig. 5.5 in red colour.

Therefore, the Cartesian coordinates of a point  $P^D$  in the referential L, are given by the following expression:

$$P^L = R_D^L \cdot P^D + T_D^L \quad (5.4)$$

The referential attached to the AX-12 shaft is shown in the figure Fig. 5.6 in blue colour. The figure shows the tilt and the laser (L) frames.

The  $y$  axis of the tilt and laser frames are coincident. The rotation of the laser referential is performed around the  $y_L$  axis with the pitch angle ( $\theta$ ). This rotation is represented by the matrix  $R_L^{Tilt}$ , and is expressed in the tilt referential coordinates:

$$R_L^{Tilt} = \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \quad (5.5)$$

The translation of the laser frame, in relation to the tilt frame, is given by the vector  $T_L^{Tilt}$ . This translation vector is written as coordinates of the tilt referential.

$$T_L^{Tilt} = [0 \quad -T_{y_{Tilt,L}} \quad 0]^T \quad (5.6)$$

in which the distance value  $T_{y_{Tilt,L}}$ , is represented in red in Fig. 5.6.

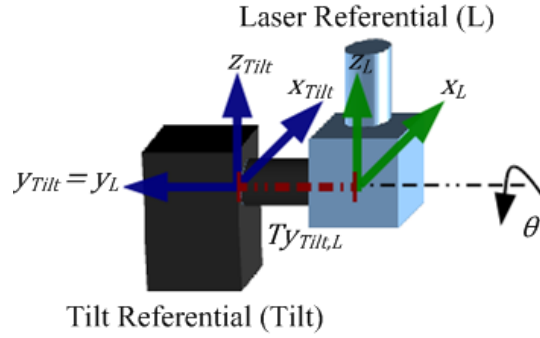


Fig. 5.6 Tilt referential (Tilt) in blue. Laser referential (L) in green.

Therefore, a point in the laser referential can be represented in the tilt referential, using the following equation:

$$p^{Tilt} = R_L^{Tilt} \cdot p^L + T_L^{Tilt} \quad (5.7)$$

The Fig. 5.7 represents the vehicle referential (V) in brown. This referential is in the middle of the connection between the two traction wheels of the robot. The tilt referential is represented in blue. Finally, the red line shows the vector  $T_{Tilt}^V$ .

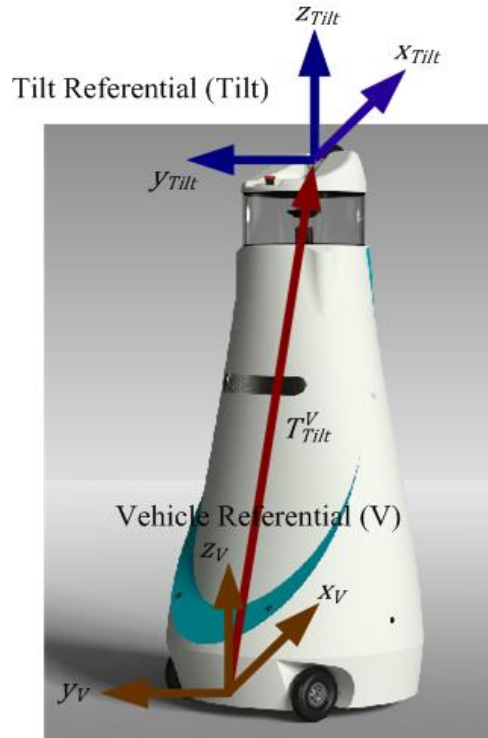


Fig. 5.7 Vehicle referential (V) at brown, attached to the vehicle odometry's referential. Tilt referential (Tilt) in blue.

The vector  $T_{Tilt}^V$  represents the translation vector of the tilt referential, in relation to the V referential, expressed in coordinates of V. This vector can be expressed as the following equation:

$$T_{Tilt}^V = [T_{x_{V,Tilt}} \quad T_{y_{V,Tilt}} \quad T_{z_{V,Tilt}}]^T \quad (5.8)$$

A point in the tilt referential can be represented in the vehicle referential with the following expression:

$$P^V = R_{Tilt}^V \cdot P^{Tilt} + T_{Tilt}^V \quad (5.9)$$

Finally, the entire transformation of the points  $P^D$  on the frame  $D$  to the vehicle referential (V), is obtained using the expression:

$$P^V = R_{Tilt}^V \cdot R_L^{Tilt} \cdot R_D^L \cdot P^D + R_{Tilt}^V \cdot R_L^{Tilt} \cdot T_D^L + R_{Tilt}^V \cdot T_L^{Tilt} + T_{Tilt}^V \quad (5.10)$$

As there is no rotation of the tilt referential, in relation to the vehicle referential, the matrix  $R_{Tilt}^V$  is the identity matrix.

Consider now that the vehicle is well located and its position in the world frame is known. Therefore, the point  $P^V$  corresponding to the measurement  $P^D$  can also be represented in the world frame.

In Fig. 5.8 it is shown the vehicle referential (V) in brown and the world frame (W) in black. The vehicle x axis, placed in the origin of referential W is represented by a dashed brown vector.

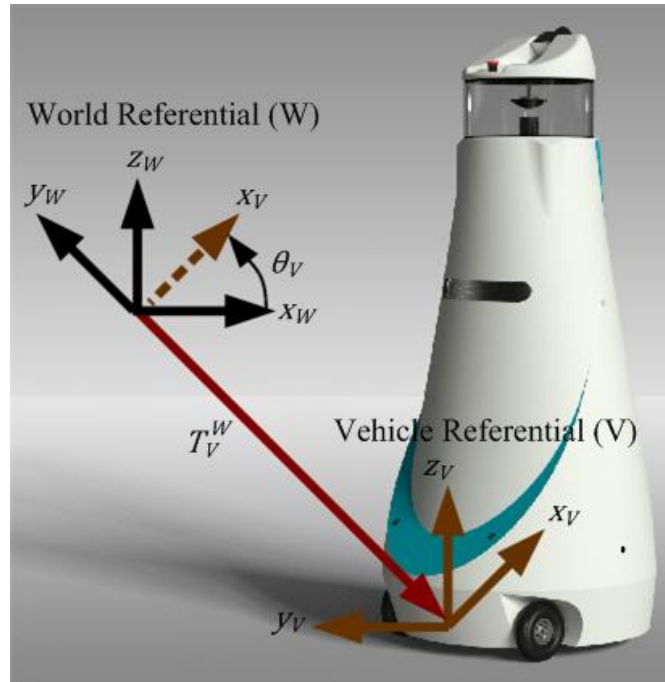


Fig. 5.8 Vehicle referential (V) in brown. World referential (W) in black.

The translation vector between the vehicle and world frames, expressed in the coordinates of the world frame, is represented by the vector  $T_V^W$  and, in Fig. 5.8, is shown in red. This vector is equal to the following expression:

$$T_V^W = [T_{x_{W,V}} \quad T_{y_{W,V}} \quad 0]^T \quad (5.11)$$



The vehicle pose estimation is equal to the vehicle referential position and orientation in relation to the world frame. Therefore, the position of the referential V, in the referential W, is equal to the vector  $T_V^W$ .

The vehicle rotation  $\theta_V$  is the rotation of the referential V, in relation to the referential W, represented in the world frame by the matrix  $R_V^W$ . This rotation is performed around the  $z_V$  axis and is equal to the following matrix:

$$R_V^W = \begin{bmatrix} \cos \theta_V & -\sin \theta_V & 0 \\ \sin \theta_V & \cos \theta_V & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.12)$$

Therefore, the point  $P^V$  corresponding to the measurement  $P^D$ , can be written in the world referential, using the following expression:

$$P^W = R_V^W \cdot P^V + T_V^W \quad (5.13)$$

Finally, a measurement  $P^D$  can be re-written as a point  $P^W$  with the following equation:

$$P^W = R_V^W \cdot [R_{Tilt}^V \cdot R_L^{Tilt} \cdot R_D^L \cdot P^D + R_{Tilt}^V \cdot R_L^{Tilt} \cdot T_D^L + R_{Tilt}^V \cdot T_L^{Tilt} + T_{Tilt}^V] + T_V^W \quad (5.14)$$

The translation vectors  $T_D^L$ ,  $T_L^{Tilt}$  and  $T_{Tilt}^V$  are fixed parameters on the observation module position on the robot. These values were carefully measured. The vector  $T_V^W$  and angle  $\theta_V$  is obtained with the vehicle localization algorithm, *three-dimensional map-based* approach, described in the following chapters of this PhD document.

The Laser Range Finder has a built-in a laser beam rotating at a speed of 60 rpms. The laser rotates the full laser angle range (360°) in 100 milliseconds (the Laser Range Finder time cycle). After each rotation (360°), the LRF sends 769 measures. The first reading (0<sup>th</sup>) corresponds to the minimum yaw,  $\Psi_{min} = -135.19^\circ$ . The 768<sup>th</sup> reading corresponds to the maximum yaw,  $\Psi_{max} = 135.19^\circ$ . The laser reading  $i$  has an angle  $\Psi_i$ , equal to the following expression:

$$\psi_i = \frac{(\Psi_{max} - \Psi_{min})}{768} \cdot i + \Psi_{min}; i = 0, \dots, 768 \quad (5.15)$$

At each laser sample rate there is a reading of the AX-12 present angle ( $Pres_{Ang}$ ) and present speed ( $Prest_{RPM}$ ). Due the rotation of the AX-12 during the complete scan of the Laser Range Finder, the acquired readings will represent a curve as shown in Fig. 5.9. Therefore, the roll angle ( $\theta_i$ ) must be corrected for each reading  $i$ .

The Hokuyo LRF is composed by a servo motor inside, which puts a laser beam in rotation, performing the entire scanning. Each entire rotation of the laser beam (360°), comprises 1024 steps. After each rotation of 1024 steps, the LRF sends only 769 measures, referent to the angles between  $\Psi_{min}$  and  $\Psi_{max}$ . The first reading, corresponding to the yaw angle  $\Psi_{min}$ , has a corresponding step equal to:

$$step_0 = \frac{1024 - 768}{2} \quad (5.16)$$

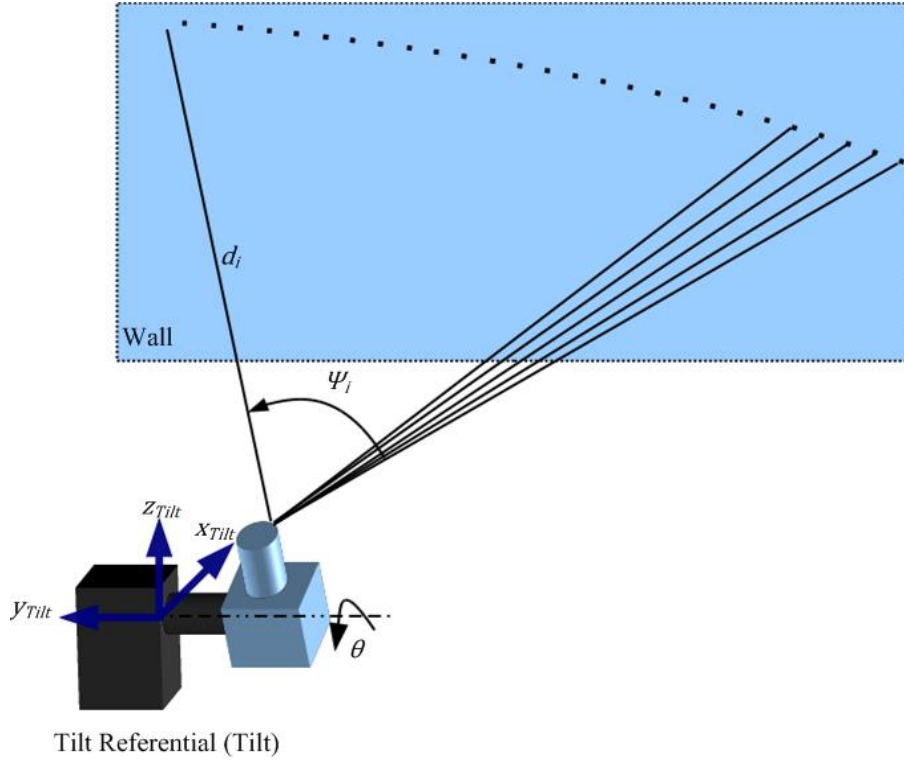


Fig. 5.9 Curvature on the readings caused by the AX12-12 rotation.

While the last reading, corresponding to the yaw angle  $\Psi_{\max}$ , as a corresponding step equal to:

$$step_{768} = \frac{1024 + 768}{2} \quad (5.17)$$

The reading  $i$  ( $i^{\text{th}}$ ) has a corresponding step  $i$ . Therefore, the reading  $i$  has a corresponding step equal to the following expression:

$$step_i = \frac{1024 - 768}{2} + i = 128 + i; i = 0, \dots, 768 \quad (5.18)$$

As the Laser Range Finder has a sample rate of 100 milliseconds, the time between each laser step is equal to:

$$\Delta t_{step} = \frac{0.1}{1024} \text{ (seconds)} \quad (5.19)$$

The reading of the servo motor present speed,  $Pres_{RPM}$ , has units equal to rpms. The same speed but in radians/second can be computed using the following expression:

$$Pres_{rad/sec} = \frac{2\pi \cdot Pres_{RPM}}{60} \text{ (radians/second)} \quad (5.20)$$

To synchronise the AX-12's angle with the data acquired by the LRF, the time between the last AX-12 data arrival and the LRF readings is counted and equal to  $\Delta T$  (seconds). Therefore, the AX-12 angle corresponding to the last step (1024), is equal to the last AX-12's

angle measurement,  $Pres_{Ang}$  (radians), plus the AX12's amount of rotation during the time  $\Delta T$ :

$$\theta_{1024} = Pres_{Ang} + cw \cdot \Delta T \cdot Pres_{rad/sec} \text{ (radians)} \quad (5.21)$$

in which the variable  $cw$  represents the direction of rotation.

Therefore, to synchronise each reading  $i$ , with the correct angle of the AX-12 servo motor, it is necessary to compute the angle corresponding to the  $step_i$ . The angle corresponding to the  $step_i$  is equal to  $\theta_{step_i}$  and is calculated by the difference between the angle of the full laser range ( $\theta_{1024}$ ); and the angle rotated by the AX-12 between the  $step_i$  and the step 1024. Therefore, the angle corresponding to the  $step_i$ , the step of the reading  $i$ , is equal to:

$$\theta_{step_i} = \theta_{1024} - cw \cdot \Delta t_{step} \cdot (1024 - step_i) \cdot Pres_{rad/sec} \text{ (radians)} \quad (5.22)$$

But, it is desired to obtain the roll angle in accordance with the reading  $i$  ( $\theta_i$ ). Then, using the equations (5.18) and (5.22) the expression of  $\theta_i$  is equal to the following expression:

$$\theta_i = \theta_{1024} - cw \cdot \Delta t_{step} \cdot (896 - i) \cdot Pres_{rad/sec} \text{ (radians); } i = 0, \dots, 768 \quad (5.23)$$

The last equation can still be re-written in a simpler way, using the equation (5.21), as shown in the expression:

$$\theta_i = Pres_{Ang} + cw \cdot Pres_{rad/sec} [\Delta T - \Delta t_{step} \cdot (896 - i)] \text{ (radians); } i = 0, \dots, 768 \quad (5.24)$$

### 5.3. Odometry

At each cycle time, the odometry data obtained, is computed through the acquired pulses of the odometer, at each traction wheel. The odometry data is represented by  $Odo$  and given by the following vector:

$$Odo = \begin{bmatrix} d \\ \Delta\theta \end{bmatrix} \quad (5.25)$$

in which  $d$  in metres and  $\Delta\theta$  in radians, represent the linear and angular displacement between two consecutive time steps. These values can be obtained through the following equations:

$$d = \frac{d_1 + d_2}{2} \quad (5.26)$$

$$\Delta\theta = \frac{d_1 - d_2}{b} \quad (5.27)$$

in which  $d_1$  and  $d_2$  are the distance travelled by each wheel. The variable  $b$  represents the distance between the two traction wheels, as is shown in Fig. 5.10.

The distance travelled by each wheel is given by the expressions:

$$d_1 = \frac{Odo_1}{C_{odo1}}, \quad d_2 = \frac{Odo_2}{C_{odo2}} \quad (5.28)$$

in which  $Odo_1$  and  $Odo_2$  are the measurements of the odometers of each wheel, respectively (number of odometer pulses between consecutive time steps).

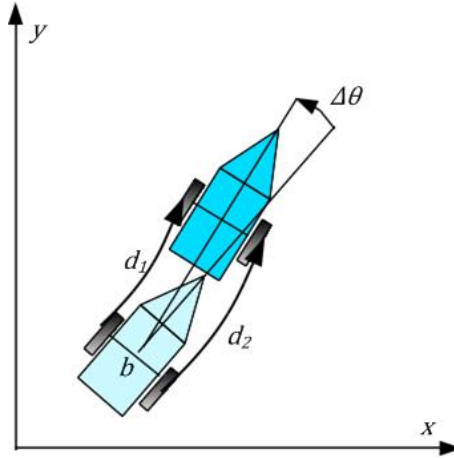


Fig. 5.10 Linear displacement of each wheel and angular rotation between two different time steps.

The variables  $C_{Odo1}$  (pulses/metre) and  $C_{Odo2}$  (pulses/metre), are constant values representing the number of odometer pulses per metre, travelled by each wheel. Therefore, the odometer constants  $C_{Odo1}$  and  $C_{Odo2}$  can be computed as follows:

$$C_{Odo1} = \frac{n}{2 \cdot \pi r_1}, \quad C_{Odo2} = \frac{n}{2 \cdot \pi r_2} \quad (5.29)$$

where  $n$  is the number of pulses of the odometer during a complete turn, while  $r_1$  and  $r_2$  are the radius values of each wheel, respectively.

The value  $n$  is equal to the encoder number of pulses per wheel rotation. This value is known and is equal to 33792 pulses.

Aiming to calibrate the odometry and compute the parameters  $r_1$ ,  $r_2$  and  $b$ , the following experiments were performed:

1) Controlling the robot with the joystick, the vehicle moved forward, during 20 tests, travelling a distance of 5 metres.

During these 20 tests, the travelled distances by each wheel were,  $d_1 = d_2 = 5$  metres. The average of the number pulses in each wheel was  $Odo_1 = 409400$  pulses and  $Odo_2 = 410780$  pulses. Therefore, the radius  $r_1$  and  $r_2$ , were computed using the equations (5.28) and (5.29):

$$r_1 = \frac{n}{2 \cdot \pi \cdot \frac{Odo_1}{d_1}}, \quad r_2 = \frac{n}{2 \cdot \pi \cdot \frac{Odo_2}{d_2}} \quad (5.30)$$

Resulting in the values of  $r_1 = 0.0657$  metres and  $r_2 = 0.0655$  metres. For simplicity reasons, it was assumed that:  $r_1 = r_2 = \frac{0.0657 + 0.0655}{2} = 0.0656$  metres.

2) Again controlling the robot with a joystick, the vehicle rotated, during 25 tests, three turns around itself. During these 25 tests, the rotated angle was  $\Delta\theta = 3 \cdot 2\pi$  radians. Using the previous computed values, for  $r_1$  and  $r_2$ , the average of the travelling distances  $d_1$  and  $d_2$  was

3.92 metres and -3.87 metres, respectively. Therefore, using the equation (5.27), the parameter  $b$  comes equal to 0.4133 metres.

These experiments were performed in an unique type of floor, with the vehicle moving at a speed of 0.2 m/s, avoiding the slippage of its traction wheels. The floor was carefully marked aiming to measure the true translation and rotation of the vehicle with a maximum accuracy possible.

J. Borenstein and L. Feng [8], suggest an experiment to correct and improve the values of  $r_1$ ,  $r_2$  and  $b$ , after performed the calibration of the odometry, such as it was done above. But, these parameters are influenced with a set of conditions and factors, which makes such correction a vain effort. Some of these factors are, for instance, the type of floor, the temperature, which, in the case of rubber wheels, has great influence in the wheels form, radius and middle contact point with the floor.

Another two experiments were performed, aiming to estimate the error of the odometry in relation to reality. Again, in these experiments, the same conditions applied above are true (vehicle moving at a speed of 0.2 m/s and markers at the floor).

In these two experiments it is considered that the error associated to the distance travelled forward, is modelled as random noise, with a normal distribution (zero mean and standard deviation equal to  $\sigma_d \cdot d$ , proportional to the distance  $d$ , acquired by odometry). The  $\sigma_d$  is the standard deviation percentage with units equal to metres/metres. Therefore, it is considered that the error in the distance travelled forward, is smaller or equal than  $\sigma_d \cdot d$ , with a certainty of 68.2%.

It is also assumed that the error in the rotation of the vehicle is modelled as random noise, with a normal distribution (zero mean and standard deviation  $\sigma_{d\theta} \cdot d + \sigma_\theta \cdot \Delta\theta$ , proportional to the distance  $d$  and angle  $\Delta\theta$ , measured by odometry). The parameters  $\sigma_{d\theta}$  and  $\sigma_\theta$  are percentage standard deviations, with units equal to radians/metres and radians/radians, respectively. Then, it is assumed that the error of the angle estimative, using odometry, is lower than  $\sigma_{d\theta} \cdot d + \sigma_\theta \cdot \Delta\theta$ , with an certainty of 68.2%.

Therefore, with the following two experiments, it is intended to estimate the values of  $\sigma_d$ ,  $\sigma_{d\theta}$  and  $\sigma_\theta$  in the following manner:

1) Controlling the robot with a joystick, the vehicle moved forward for 20 times a distance of 5 metres. During each test, the odometry computed the travelled distance. The average of the ratio between the odometry estimative of the travelled distance and the real travelled distance (5 metres), allows obtaining the standard deviation  $\sigma_d$ .

Also, the average ratio between the estimated rotation using the odometry and the travelled distance in front (5 metres) allows determining  $\sigma_{d\theta}$ . The obtained results were  $\sigma_d = 0.18264$  metres/metres and  $\sigma_{d\theta} = 0.08961$  radians/metres.

2) Again controlling the robot with a joystick, the vehicle performed 20 times, 3 complete rotations. During each time it was computed, using the odometry, the amount of rotation. The average ratio between the rotation estimated with the odometry and the real ( $\Delta\theta = 3 \cdot 2\pi$  radians), makes it possible to compute the standard deviation  $\sigma_\theta$ . The obtained result was:  $\sigma_\theta = 0.02819$  radians/radians.

The results obtained with the odometry calibration are summarised in the following table.

$r_1 = r_2$ (metres)	$b$ (metres)	$\sigma_d$ (metres/metres)	$\sigma_{d\theta}$ (radian /metres)	$\sigma_\theta$ (radians/radians)
0.0656	0.4160	0.18264	0.08961	0.02819

Table 5.1 Result values of the odometry calibration.

## 6. Data Detection and Extraction

So as to perform the *pre-localization* of the *three dimensional map-based* approach, using the SLAM procedure and build a 2D feature map, the features on the environment have to be extracted through laser data. This chapter describes the Data Detection and Extraction process, aiming to find linear segments (representing walls, doors, furniture *et cetera*) and invariant points (representing corners or circular columns), in the vehicle frame.

The tilting LRF, *observation module*, acquires  $P$  points in the 3D space, with coordinates  $(x, y, z)$ . Through the points  $P$  it is possible to obtain the set of points  $P_{2D}$ , which are the projection in the two-dimensional space, of the points  $P$  with  $z$  coordinates higher than  $Z_{min}$  metres and lower than  $Z_{max}$ . The values of the parameters  $Z_{min}$  and  $Z_{max}$  are presented in the Chapter of results, Chapter 10.

As the *pre-localization* is performed in a controlled scenario, without dynamic objects or people moving in the navigation area, the  $P_{2D}$  points are used in the Data Detection and Extraction module, in order to obtain the linear segments and invariant points used in the *pre-localization*.

All the points and parameters used and presented in this Chapter are represented in the vehicle referential. Therefore, in this Chapter, the superscript index referring to the representation on the vehicle referential shall be omitted.

### 6.1. Data Detection

In the following paragraphs a set of important definitions are given. These definitions are crucial in the data detection algorithm described herein.

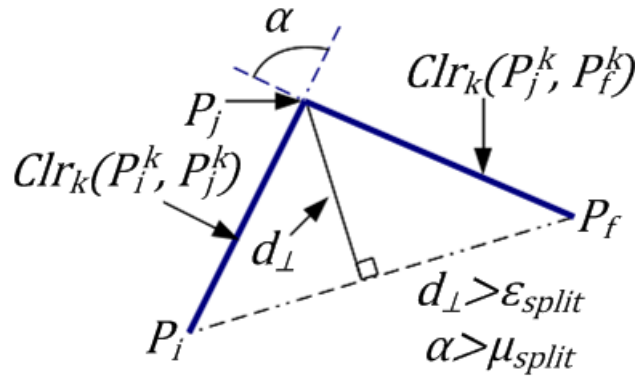


Fig. 6.1 Measures  $d_{\perp}$  and  $\alpha$ .

**Definition 1** The list of points  $P_{2D}$  is represented with the vector  $[P_1, \dots, P_j, \dots, P_n]$ .

**Definition 2** As shown in Fig. 6.1,  $Clr_k$  is a list of connected points, called "cluster", having as initial and final points  $P_i^k$  and  $P_f^k$ .

**Definition 3**  $Lin_k$  is a list of connected points, called "linear segment", with extreme points  $P_i^k$  and  $P_f^k$ .

**Definition 4**  $Circ_k$  is a list of connected points, called "circular segment", with extreme points  $P_i^k$  and  $P_f^k$ .

**Definition 5** The value  $d_\perp$  is the perpendicular distance between  $P_j^k \in Clr_k$  and the segment line, which connects  $P_i^k$  to  $P_f^k$ .

**Definition 6** The value  $\alpha$  is the orientation difference between  $Clr_k(P_i^k, P_j^k)$  and  $Clr_k(P_j^k, P_f^k)$ , as represented in Fig. 6.1. In which  $Clr_k(P_i^k, P_j^k)$  is the segment line that connects  $P_i^k$  and  $P_j^k$ , while the  $Clr_k(P_j^k, P_f^k)$  is the segment line that connects the points  $P_j^k$  and  $P_f^k$ .

**Definition 7** The cluster  $Clr_k$  can be considered a candidate for a linear segment or a set of candidates for linear segments. In Fig. 6.2, an example of a cluster representing a set of candidates for linear segments is  $Clr_3$ . On contrary the cluster  $Clr_k$  can be considered a circular segment candidate. In Fig. 6.2, an example of a candidate for a circular segment is the cluster  $Clr_4$ .

**Definition 8** We shall consider now the equation of a circular segment (circle) as the following, with parameters relatively to the vehicle frame:

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \quad (6.1)$$

in which the point  $(x_0, y_0)$  is the circle's centre coordinates in the vehicle referential, with radius  $r$ . The circle segment equation can be re-written to become a linear system:

$$\begin{aligned} x^2 + y^2 - 2xx_0 - 2yy_0 + x_0^2 + y_0^2 &= r^2 \Leftrightarrow \\ x^2 + y^2 &= 2xx_0 + 2yy_0 + x_0^2 + y_0^2 + r^2 \Leftrightarrow \\ (x^2 + y^2) &= [x \quad y \quad 1] \begin{bmatrix} 2x_0 \\ 2y_0 \\ r^2 + x_0^2 + y_0^2 \end{bmatrix} \end{aligned} \quad (6.2)$$

Therefore, the circle segment equation was transformed in a linear system, which can be re-written with the following expressions:

$$Y_{Circ} = \begin{bmatrix} x_i^2 + y_i^2 \\ \dots \\ x_f^2 + y_f^2 \end{bmatrix}, X_{Circ} = \begin{bmatrix} x_i & y_i & 1 \\ \dots & \dots & \dots \\ x_f & y_f & 1 \end{bmatrix}, g_{Circ} = \begin{bmatrix} 2x_0 \\ 2y_0 \\ r^2 + x_0^2 + y_0^2 \end{bmatrix} \quad (6.3)$$

$$Y_{Circ} = X_{Circ} g_{Circ} \quad (6.4)$$

in which  $x_j$  and  $y_j$  are the coordinates of the point  $P_j^k \in Clr_k$ .

The classic Least Square Quadratic Error (LSQ) tries to find the best parameters  $g_{Circ}$  defined in equation (6.3), which minimizes the following cost function:

$$f_{Circ} = \sum_{j=i}^f \left( (x_j^2 + y_j^2) - (2x_jx_0 + 2y_jy_0 + x_0^2 + y_0^2 + r^2) \right)^2 \quad (6.5)$$

Solving the linear system (6.4), using the classic Least Square Quadratic Error (LSQ), the parameters  $g_{Circ}$  are computed as follows:



$$g_{Circ} = (X_{Circ}^T X_{Circ})^{-1} X_{Circ}^T Y_{Circ} \quad (6.6)$$

**Definition 9** The fitting error (variance) of a circular segment  $\sigma_{Circ}^2$ , is computed as follows:

$$\sigma_{Circ}^2 = \frac{\sum_{j=i}^f \left( r - \sqrt{(x_0 - P_{j,x})^2 + (y_0 - P_{j,y})^2} \right)^2}{f - i} \quad (6.7)$$

where  $f$  and  $i$  are the final and initial indexes of the cluster  $Clr_k$ .

**Definition 10** The cluster  $Clr_k$  is considered a circular segment (circle) if and only if:

1) The matrix  $X_{Circ}^T X_{Circ}$  is invertible. To determine if the matrix  $X_{Circ}^T X_{Circ}$  is invertible, a test of its determinant, equal to  $\det(X_{Circ}^T X_{Circ})$ , is conducted so as known whether it is well conditioned;

2) The radius parameter resulting from the solution of the equation (6.6) falls within the gap  $r \in [r_{min}, r_{max}]$ . In which  $r_{min}$  and  $r_{max}$  are the minimum and maximum intervals that the circle's radius can represent (depending on the radius of the circular columns in the scenario).

3) The fitting error  $\sigma_{Circ}^2$  is lower than the parameter  $\sigma_{MaxCirc}^2$ .

**Definition 11** If the entire set of the conditions on Definition 10 are not met, the cluster  $Clr_k$  is considered a candidate for a linear segment or a set of candidates for linear segments.

### 6.1.1. Merge and Split

The detection method is a Merge and Split algorithm, [57]. The algorithm described here has the advantage of extracting linear segments (from walls, doors or furniture) and invariant points (from corners or circular columns), at the same time. The algorithm is described in the following two steps:

1) *Merge*: to the entire set of points  $P_{2D}$  is tested if consecutive points  $P_j$  and  $P_{j+1}$  have an Euclidian distance lower than  $\varepsilon_{merge}$ . If this happens,  $P_j$  and  $P_{j+1}$  are merged in the same cluster. The merge result is a list of clusters  $Clr_k$ , as shown in Fig. 6.2.

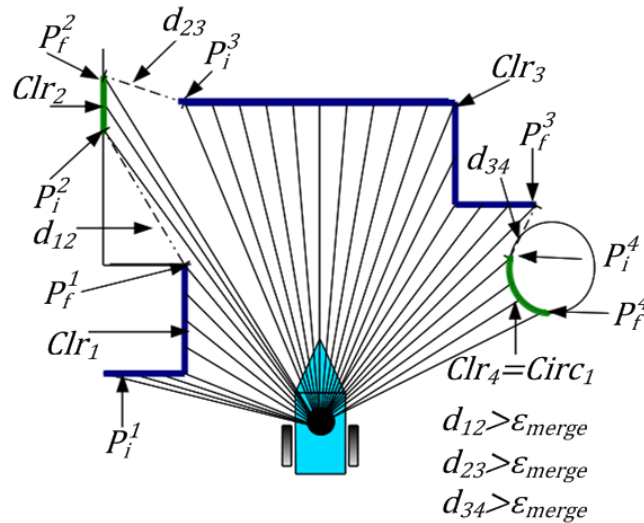


Fig. 6.2 Merge find clusters  $Clr_{P_i,f}$  and circular segments  $Circ_{P_i,f}$ .

Through Definition 10 and Definition 11 each cluster  $Clr_k$  is classified as a circular segment or a set of linear segment candidates. To each cluster,  $Clr_k$ , classified as a set of linear segment candidates, the split phase is applied.

Linear segment candidates are shown in Fig. 6.2 with  $Clr_1$ ,  $Clr_2$  and  $Clr_3$ . A circular segment candidate is shown at Fig. 6.2 with  $Clr_4$ .

2) *Split*: each cluster  $Clr_k$ , classified as candidates for linear segments, the point  $P_j^k \in Clr_k$  is searched, whose  $d_{\perp}$  is the greatest.  $Clr_k$  is iteratively divided in clusters  $Clr_k(P_i^k, P_j^k)$  and  $Clr_k(P_j^k, P_f^k)$ , as represented in Fig. 6.1, while the following conditions are verified:

- i)  $d_{\perp}$  is greater than  $\varepsilon_{split}$ .
- ii)  $\alpha$  is lower than  $\mu_{split}$ .

When the sub-cluster  $Clr_k$  ceases to be divided, it is considered a linear segment  $Lin_j$ . The result is shown in Fig. 6.3.

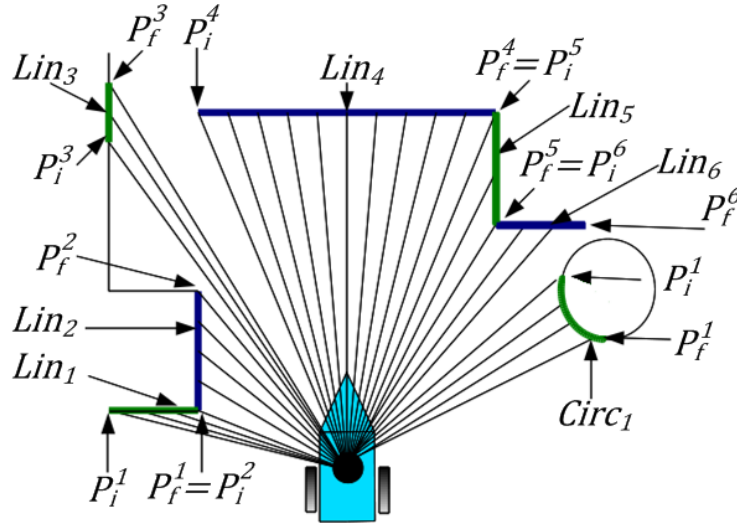


Fig. 6.3 Split find linear segments  $Lin_{pi,f}$

### 6.1.2. Invariant Points Classification

After applying the merge and split algorithm, the result is a list of linear segments  $Lin_j$  and circular segments  $Circ_j$  with extremes equal to  $P_i^j$  and  $P_f^j$ .

The centre of the circular segments is considered an invariant point, since the centre of a circle is invariant with the vehicle displacement.

In the case of a linear segment  $Lin_j$ , its extreme points are tested to determine whether these are invariant points. Consider the segments  $Lin_j$  and  $Lin_{j+1}$  with extreme points  $P_i^j$  and  $P_f^j$ ,  $P_i^{j+1}$  and  $P_f^{j+1}$ , respectively. The points  $P_f^j$  and  $P_i^{j+1}$  have distance readings equal to  $R_f^j$  and  $R_i^{j+1}$ , respectively. The angle distance between them is  $\theta_{f,i}^{j,j+1}$ , as shown in Fig. 6.4.

The points  $P_f^j$  and  $P_i^{j+1}$  are considered invariant points, or not, according to the following conditions: 1) both,  $P_f^j$  and  $P_i^{j+1}$ , are invariant points if their Euclidian distance is lower than  $\varepsilon_{max}$ , therefore the resultant invariant point is the weighed average of  $P_f^j$  and  $P_i^{j+1}$ ; 2) if the

condition 1) is not met,  $P_f^j$  is an invariant point while  $P_i^{j+1}$  is not, when the following three conditions are met at the same time:

i)  $R_f^j$  and  $R_i^{j+1}$  are lower than 5 metres (LRF range).

ii)  $R_i^{j+1} > R_f^j$ .

iii) The angle  $\theta_{f,i}^{j,j+1}$  is below  $\theta_{max}$ .

In Fig. 6.4 and Fig. 6.5, the detected linear segments are shown in blue and green colours. The invariant points are represented in a square shape. The ones which meet the condition 1) are in red. The others which meet the condition 2) are in brown.

The result of the detection module is a set of linear segments  $S_{2D}$  and invariant points  $I_{2D}$ , whose parameters and covariance error are extracted in the data extraction module.

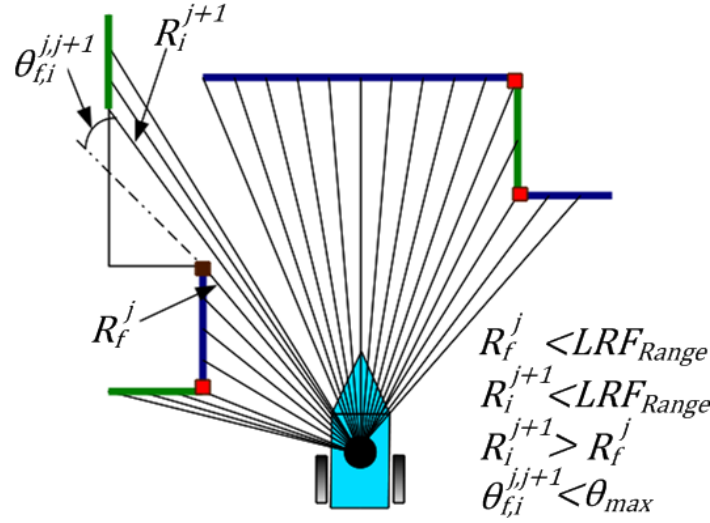


Fig. 6.4 Invariant points in a squared red and brown shape. The ones which meet the condition 1) are in red. The others which meet the condition 2) are in brown.

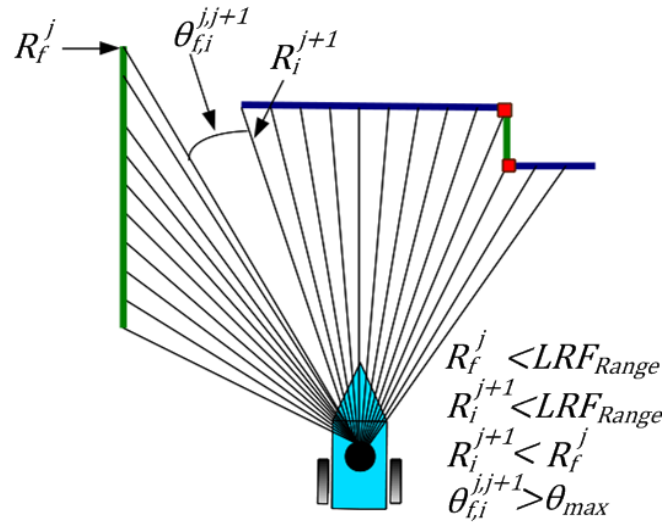


Fig. 6.5 Invariant points classification. The ones which meet the condition 1) are in red. The conditions defined above does not occur in the case of the points  $P_f^j$  and  $P_i^{j+1}$ .

### 6.1.3. Results and Adjustable Parameters

The Table 6.1 shows the parameters of the merge and split method ( $\epsilon_{\text{merge}}$ ,  $\epsilon_{\text{MaxCirc}}$ ,  $\epsilon_{\text{split}}$ ,  $\mu_{\text{split}}$ ). Table 6.2 presents the parameters of the invariant points classification method ( $\epsilon_{\text{max}}$ ,  $\theta_{\text{max}}$ ).

$\epsilon_{\text{merge}}$ (metres)	0,061	$\epsilon_{\text{split}}$ (metres)	0,061
$\sigma_{\text{Maxcirc}}^2$ (meters <sup>2</sup> )	0.001	$\mu_{\text{split}}$ (°)	170

Table 6.1 Parameters of the Merge and split method.

$\epsilon_{\text{max}}$ (metres)	0,061
$\theta_{\text{max}}$ (°)	3,5

Table 6.2 Parameters of the invariant points classification method.

The parameters  $\epsilon_{\text{merge}}$ ,  $\epsilon_{\text{split}}$  and  $\epsilon_{\text{max}}$  are equal to 0.061 metres. Considered together with the Laser Range Finder's angle resolution ( $0.35^\circ$ ) and its maximum distance range (5 metres), the size of the arc between consecutive readings is given by the following equation:

$$s = d \cdot \theta \Leftrightarrow$$

$$s = 5 \cdot \frac{0.35}{180} \pi \approx 0.0305 \text{ metres} \quad (6.8)$$

The distance between consecutive readings of the LRF will be lower than the arc  $s$ . Therefore, it was decided to consider the values  $\epsilon_{\text{merge}} = \epsilon_{\text{split}} = \epsilon_{\text{max}} = 2 \cdot s = 0.061$  metres.

The parameter  $\mu_{\text{split}}$  means the minimum angle between two clusters  $Clr_k(P_i^k, P_j^k)$  and  $Clr_k(P_j^k, P_f^k)$ , as represented in Fig. 6.1, need to be divided into two clusters. It is common sense that two clusters with a  $\mu_{\text{split}}$  angle, lower than  $170^\circ$  degrees are clusters representative of two different walls, i.e. linear segments. Therefore,  $\mu_{\text{split}} = 170^\circ$ .

Bearing in mind that in case of an invariant point, marked in brown, in Fig. 6.4, the points  $P_f^j$  and  $P_i^{j+1}$  are consecutive readings of the Laser Range Finder. Therefore, as the LRF has an angular resolution of  $0.35^\circ$ , the angle between the points is  $\theta_{f,i}^{j,j+1} = 0.35^\circ$ . Therefore, the chosen value for  $\theta_{\text{max}}$  was ten times the LRF's angular resolution,  $\theta_{\text{max}} = 3.5^\circ$ .

The parameter  $\sigma_{\text{Maxcirc}}^2$  was tested in an interval of  $[10^{-4}, 10^{-2}] \text{ metres}^2$ . The best behaviour was found when  $\sigma_{\text{Maxcirc}}^2 = 0.001 \text{ metres}^2$ . The best behaviour was considered when the algorithm detected the true circular segments (columns) as circular segments and the true linear segments as linear segments, by using logged data for three different scenarios, where circular columns and walls are found.

The parameters presented at Table 6.1 and Table 6.2 proved to guarantee the best performance, while identifying correctly the entire set of present features, most of the times, using logged data for three different scenarios. The Fig. 6.6 and Fig. 6.7 show the result of the application of the Merge and Split algorithm with real data. The green lines are the detected linear segments. The green squares are invariant points, while the green circles are variant points. Finally, in Fig. 6.7, the red circles are columns.

The invariant points are points whose position is invariant with the vehicle movement and which are used as a feature during the *pre-localization*. On the contrary, the variant points are moving points with the vehicle displacement. Those are not used in the *pre-localization* procedure.

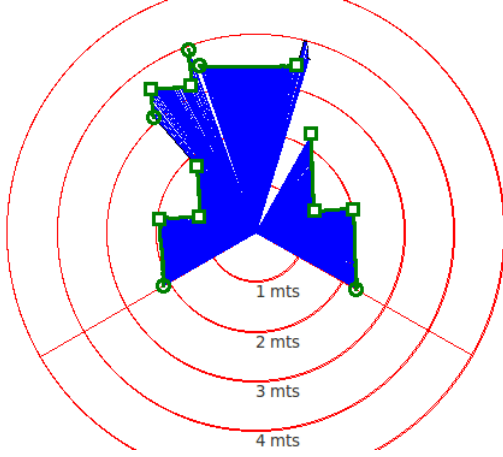


Fig. 6.6 The Merge and Split method found linear segments (green). Invariant points represented with a green squared shape, while the variant points are represented with a green circled shape.

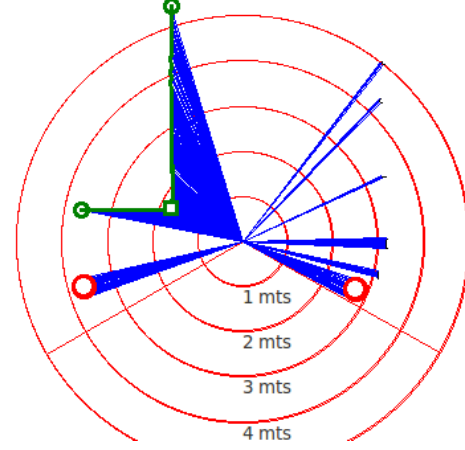


Fig. 6.7 Application of the Merge and Split. To columns are found and represented in a red circled shape.

## 6.2. Data Extraction

After the linear segments and invariant points have been detected, it is necessary to extract information from them. As the linear segments and invariant points of corners or circular segments are used during the *pre-localization* stage, Chapter 7, in an Extended Kalman Filter applied in the SLAM problem, it is necessary to extract information on their parameters and the respective covariance matrices.

The Least Square Quadratic error (LSQ) is used to obtain the linear segment parameters and covariances, as is described in [44]. The work described in [59] presents a particular and interesting approach to determining the invariant points and the respective errors with exact precision.

### 6.2.1. Linear Segment Extraction

Consider the linear segment  $Lin_{P_{i,f}}$  with points  $[P_i, \dots, P_j, \dots, P_f]$ , obtained by the tilting Laser Range Finder, with coordinates  $P_j = [x_j, y_j]$ , in the vehicle frame. The matrices  $Y_{Lin}$  and  $X_{Lin}$ , are equal to the following:

$$Y_{Lin} = [y_i \quad \dots \quad y_f]^T \quad (6.9)$$

$$X_{Lin} = \begin{bmatrix} x_j & \dots & x_f \\ 1 & \dots & 1 \end{bmatrix}^T \quad (6.10)$$

The linear segment parameters in the Cartesian representation  $y_j = kx_j + c$ , with slope  $k$  and interception with the  $y$  axis equal to  $c$ , as shown in Fig. 6.8, are given by the vector:

$$g_{k,c} = \begin{bmatrix} k \\ c \end{bmatrix} \quad (6.11)$$

Therefore, the line parameters can be re-written as a linear system, depending on the matrix  $Y_{Lin}$  and  $X_{Lin}$ , as presented in the following equation:

$$Y_{Lin} = X_{Lin} g_{k,c} \quad (6.12)$$

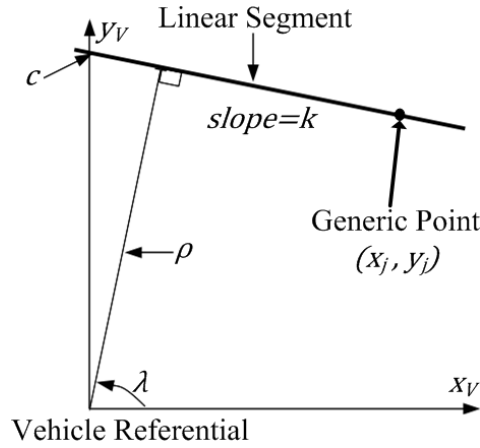


Fig. 6.8 Linear segment referring to the vehicle referential.

As the system of the previous equation is a linear system, the classic Least Square Quadratic Error (LSQ) can be applied to find the best linear segment parameters. Thus, the previous equation can be re-written as follows, while allowing to obtain an estimation on the line parameters in the Cartesian form,  $g_{k,c}$ :

$$X_{Lin}^T Y_{Lin} = X_{Lin}^T X_{Lin} g_{k,c} \Leftrightarrow$$

$$g_{k,c} = [X_{Lin}^T X_{Lin}]^{-1} X_{Lin}^T Y_{Lin} \quad (6.13)$$

The application of the LSQ in order to find the best fitting for the line parameters, minimises the following cost function:

$$f_{Lin} = \sum_{j=i}^f (y_j - \hat{y}_j)^2 \quad (6.14)$$

in which  $\hat{y}_j$  is given by the equation  $\hat{y}_j = kx_j + c$ , using the estimation of the line parameters  $g_{k,c}$ . Therefore, it is possible to define the estimated matrix  $\hat{Y}_{Lin}$ :

$$\hat{Y}_{Lin} = [\hat{y}_i \quad \dots \quad \hat{y}_f]^T \Leftrightarrow$$

$$\hat{Y}_{Lin} = [kx_i + c \quad \dots \quad kx_f + c]^T \quad (6.15)$$

The variance of the vector  $Y_{Lin}$  is given by the following equation:

$$var(Y_{Lin}) = \frac{f_{Lin}}{f - i} \quad (6.16)$$

in which  $f$  and  $i$  are the final and initial linear segment indexes, respectively, and  $f_{Lin}$  is given by the equation (6.14).

The matrix  $g_{k,c}$  can be described as function of  $Y_{Lin}$  using the Taylor expansion around the estimated vector  $\hat{Y}_{Lin}$ , as the following equation:

$$g_{k,c}(Y_{Lin}) = g_{k,c}(\hat{Y}_{Lin}) + \nabla g_{k,c}(Y_{Lin} - \hat{Y}_{Lin}) + \text{high order terms} \quad (6.17)$$

Neglecting the higher order terms,  $g_{k,c}$  as function of  $Y_{Lin}$  is given approximately by the equation:

$$g_{k,c}(Y_{Lin}) \approx g_{k,c}(\hat{Y}_{Lin}) + \nabla g_{k,c}(Y_{Lin} - \hat{Y}_{Lin}) \quad (6.18)$$

Therefore, the variance of  $g_{k,c}$ , defined as  $G_{k,c}$ , can be calculated with the expression:

$$\begin{aligned} G_{k,c} &\approx E \left[ (g_{k,c}(Y_{Lin}) - g_{k,c}(\hat{Y}_{Lin})) (g_{k,c}(Y_{Lin}) - g_{k,c}(\hat{Y}_{Lin}))^T \right] \Leftrightarrow \\ G_{k,c} &\approx E \left[ \left( \nabla g_{k,c}(Y_{Lin} - \hat{Y}_{Lin}) \right) \left( \nabla g_{k,c}(Y_{Lin} - \hat{Y}_{Lin}) \right)^T \right] \Leftrightarrow \\ G_{k,c} &\approx E \left[ \nabla g_{k,c}(Y_{Lin} - \hat{Y}_{Lin}) (Y_{Lin} - \hat{Y}_{Lin})^T \nabla g_{k,c}^T \right] \Leftrightarrow \\ G_{k,c} &\approx \nabla g_{k,c} E \left[ (Y_{Lin} - \hat{Y}_{Lin}) (Y_{Lin} - \hat{Y}_{Lin})^T \right] \nabla g_{k,c}^T \Leftrightarrow \\ G_{k,c} &= \nabla g_{k,c} \text{var}(Y_{Lin}) \nabla g_{k,c}^T \end{aligned} \quad (6.19)$$

in which  $E[.]$  is the expected value of  $[.]$ . The gradient of  $g_{k,c}$  in order to the  $Y_{Lin}$  is given by  $\nabla g_{k,c}$  and is equal to the function:

$$\nabla g_{k,c} = \frac{\partial g_{k,c}}{\partial Y_{Lin}} \quad (6.20)$$

Using the equation (6.13), the previous relation can be re-written as follows:

$$\nabla g_{k,c} = [X_{Lin}^T X_{Lin}]^{-1} X_{Lin}^T \quad (6.21)$$

Therefore, the variance matrix  $G_{k,c}$ , can be re-written as the following equations:

$$G_{k,c} = [X_{Lin}^T X_{Lin}]^{-1} X_{Lin}^T \text{var}(Y_{Lin}) \left( [X_{Lin}^T X_{Lin}]^{-1} X_{Lin}^T \right)^T \Leftrightarrow \quad (6.22)$$

$$G_{k,c} = [X_{Lin}^T X_{Lin}]^{-1} \text{var}(Y_{Lin}) \quad (6.23)$$

The linear segment equations are initially computed in the Cartesian form, because, in this situation, the dependence between the parameters and the tilting LRF obtained points can be written as a linear system, (6.12). However, to use these parameters during the *pre-localization* procedure, it is necessary to avoid the problem of the infinite slope  $k$ . Therefore, the linear segment needs to be transformed into a polar representation.

The linear segment parameters, in the vehicle frame, in the polar form, are given by the vector and expression presented in the following two equations:

$$g_{\rho,\lambda} = \begin{bmatrix} \rho \\ \lambda \end{bmatrix} \quad (6.24)$$

$$\rho = x_j \cos \lambda + y_j \sin \lambda \quad (6.25)$$

in which  $\rho$  is the perpendicular distance relative to the origin of the vehicle frame and  $\lambda$  is the angle between the perpendicular linear segment to the line and the  $x_V$  axis, as shown in Fig.

6.8. The generic point  $(x_j, y_j)$  is also represented in the figure. The parameter  $\rho$  is given with the expression:

$$\begin{aligned}\rho &= \min_j \left( \sqrt{x_j^2 + y_j^2} \right) \Leftrightarrow \\ \rho &= \min_j \left( \sqrt{x_j^2 + (k \cdot x_j + c)^2} \right) \Leftrightarrow \\ \rho &= \min_j \left( \sqrt{x_j^2(k+1) + 2k \cdot c \cdot x_j + c^2} \right)\end{aligned}\quad (6.26)$$

The solution of the equation (6.26) is obtained when  $\frac{df}{dx_j} = 0$ , in which  $f$  is equal to the following equation:

$$f = x_j^2(k+1) + 2k \cdot c \cdot x_j + c^2 \quad (6.27)$$

The solution of  $\frac{df}{dx_j} = 0$  appears when  $x_j$  and  $y_j$  are equal to the following expressions:

$$x_j = -\frac{k \cdot c}{k^2 + 1}, \quad y_j = \frac{c}{k^2 + 1} \quad (6.28)$$

Therefore, using the equation (6.26) and (6.28), the perpendicular distance  $\rho$  appears equal to the equation:

$$\begin{aligned}\rho &= \sqrt{\left(-\frac{k \cdot c}{k^2 + 1}\right)^2 + \left(\frac{c}{k^2 + 1}\right)^2} \Leftrightarrow \\ \rho &= \frac{c}{\sqrt{k^2 + 1}}\end{aligned}\quad (6.29)$$

The angle parameter  $\lambda$  is calculated as the following equation:

$$\begin{aligned}\lambda &= \tan^{-1} \left( \frac{\frac{c}{k^2 + 1}}{-\frac{k \cdot c}{k^2 + 1}} \right) \Leftrightarrow \\ \lambda &= \tan^{-1} \left( -\frac{1}{k} \right)\end{aligned}\quad (6.30)$$

Bearing in mind that the parameter  $\rho$  should be positive, the linear segment parameters in the polar form, represented by the matrix  $g_{\rho, \lambda}$ , can be re-written as the following vector:

$$g_{\rho, \lambda} = \begin{bmatrix} \frac{|c|}{\sqrt{k^2 + 1}} \\ \tan^{-1} \left( -\frac{\text{sign}(c)}{k} \right) \end{bmatrix} \quad (6.31)$$



in which  $sign(c)$  represents the signal of  $c$ . In order to become the LSQ method well conditioned, all the points  $P_j$  of the linear segment are previously rotated with a angle of  $-\frac{\pi}{2} \rightarrow (-y_j, x_j)$ , if the the following condition is met:

$$|y_i - y_f| > |x_i - x_f| \quad (6.32)$$

This condition occurs in linear segments with a slope higher than one. Therefore, this operation becomes the linear segment's slope lower than one, avoiding a high or infinite slope, making the LSQ application more accurate when determining the linear segments' parameters.

When the estimation method of the classic LSQ is applied with the linear segment's points rotated with an angle of  $-\frac{\pi}{2}$ , at the final, the parameter  $\lambda$  needs to be rotated with an angle of  $\frac{\pi}{2}$ .

Therefore, the variance of  $g_{\rho,\lambda}$ , defined as  $G_{\rho,\lambda}$  needs to be now written as a function of the variance  $G_{k,c}$ . The same deduction that results in the equation (6.19) allows computing the variance of  $g_{\rho,\lambda}$ , as function of the variance  $G_{k,c}$ , as the expression:

$$G_{\rho,\lambda} \approx \nabla g_{\rho,\lambda} G_{k,c} \nabla g_{\rho,\lambda}^T \quad (6.33)$$

The last equation can be simplified through the equation (6.23), specified in the following expression:

$$G_{\rho,\lambda} \approx \nabla g_{\rho,\lambda} [X_{Lin}^T X_{Lin}]^{-1} \nabla g_{\rho,\lambda}^T var(Y_{Lin}) \quad (6.34)$$

in which  $\nabla g_{\rho,\lambda}$  is the gradient of  $g_{\rho,\lambda}$ , given by the following matrices:

$$\begin{aligned} \nabla g_{\rho,\lambda} &= \begin{bmatrix} \frac{\partial \rho}{\partial k} & \frac{\partial \rho}{\partial c} \\ \frac{\partial \lambda}{\partial k} & \frac{\partial \lambda}{\partial c} \end{bmatrix} \Leftrightarrow \\ \nabla g_{\rho,\lambda} &= \begin{bmatrix} -\frac{k \cdot c}{(k^2 + 1)^{\frac{3}{2}}} & \frac{1}{k^2 + 1} \\ \frac{1}{k^2 + 1} & 0 \end{bmatrix} \end{aligned} \quad (6.35)$$

To sum up, the linear segment parameters in the polar form, in the vehicle frame, are computed through the equation (6.31), while the respective covariance matrix can be obtained through the equation (6.34):

$$\begin{aligned} G_{\rho,\lambda} &= \begin{bmatrix} \sigma_\rho^2 & \sigma_{\rho\lambda} \\ \sigma_{\rho\lambda} & \sigma_\lambda^2 \end{bmatrix} \Leftrightarrow \\ G_{\rho,\lambda} &\approx \begin{bmatrix} -\frac{k \cdot c}{(k^2 + 1)^{\frac{3}{2}}} & \frac{1}{k^2 + 1} \\ \frac{1}{k^2 + 1} & 0 \end{bmatrix} [X_{Lin}^T X_{Lin}]^{-1} \begin{bmatrix} -\frac{k \cdot c}{(k^2 + 1)^{\frac{3}{2}}} & \frac{1}{k^2 + 1} \\ \frac{1}{k^2 + 1} & 0 \end{bmatrix}^T var(Y_{Lin}) \end{aligned} \quad (6.36)$$

### 6.2.2. Circular Segment Extraction

The circular segment parameters computation was already described in sub-section 6.1. Solving the equations (6.3) and (6.4), by using the classic Least Square Quadratic Error (LSQ), minimising the equation (6.5), the parameters  $g_{circ}$  appears equal to the equation (6.6).

Hence, as the expression of equation was deduced (6.19), it is possible to state that the parameters of  $g_{circ}$  have a covariance equal to  $G_{circ}$ , given by the following equation:

$$G_{circ} = \nabla g_{circ} \text{var}(Y_{circ}) \nabla g_{circ}^T \quad (6.37)$$

in which the gradient of  $g_{circ}$ , equal to  $\nabla g_{circ}$ , in order to the  $Y_{circ}$  matrix, defined in equation (6.3), is given by the following expression:

$$\nabla g_{circ} = (X_{circ}^T X_{circ})^{-1} X_{circ}^T \quad (6.38)$$

where  $X_{circ}$  is defined also in equation (6.3).

Therefore, the variance matrix  $G_{circ}$  can be re-written as the following equation:

$$G_{circ} = (X_{circ}^T X_{circ})^{-1} \text{var}(Y_{circ}) \quad (6.39)$$

On the other hand, the variance of  $Y_{circ}$ , equal to  $\text{var}(Y_{circ})$ , is expressed as a function of the cost function defined in the equation (6.5):

$$\text{var}(Y_{circ}) = \frac{f_{circ}}{f - i} \quad (6.40)$$

in which  $f$  and  $i$  are the final and initial circular segment indexes, respectively.

We shall now consider the computed parameters  $g_{circ}$ , defined in the equation (6.3), written in new way, as in the following equation:

$$g_{circ} = [g_1 \quad g_2 \quad g_3]^T \quad (6.41)$$

in which  $g_1 = 2x_0$ ,  $g_2 = 2y_0$  and  $g_3 = r^2 + x_0^2 + y_0^2$ .

We shall define now the matrix of the circular segment parameters, equal to  $g_{x_0, y_0, r}$ :

$$g_{x_0, y_0, r} = \begin{bmatrix} x_0 \\ y_0 \\ r \end{bmatrix} = \begin{bmatrix} \frac{g_1}{2} \\ \frac{g_2}{2} \\ \sqrt{g_3 - \frac{g_1^2 + g_2^2}{4}} \end{bmatrix} \quad (6.42)$$

Therefore, the variance of  $g_{x_0, y_0, r}$ , defined as  $G_{x_0, y_0, r}$ , is calculated with the expression:

$$G_{x_0, y_0, r} \approx \nabla g_{x_0, y_0, r} G_{circ} \nabla g_{x_0, y_0, r}^T \quad (6.43)$$

The gradient of  $g_{x_0,y_0,r}$  in order to the parameters of  $g_{circ}$ , is given by the following expression:

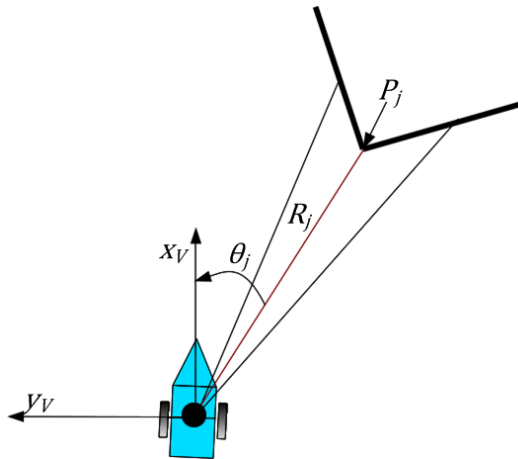
$$\begin{aligned} \nabla g_{x_0,y_0,r} &= \begin{bmatrix} \frac{\partial x_0}{\partial g_1} & \frac{\partial x_0}{\partial g_2} & \frac{\partial x_0}{\partial g_3} \\ \frac{\partial y_0}{\partial g_1} & \frac{\partial y_0}{\partial g_2} & \frac{\partial y_0}{\partial g_3} \\ \frac{\partial r}{\partial g_1} & \frac{\partial r}{\partial g_2} & \frac{\partial r}{\partial g_3} \end{bmatrix} \Leftrightarrow \\ \nabla g_{x_0,y_0,r} &= \frac{1}{2} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{x_0}{r} & -\frac{y_0}{r} & \frac{1}{r} \end{bmatrix} \end{aligned} \quad (6.44)$$

In brief, the circular segment parameters  $g_{x_0,y_0,r}$ , are obtained with the equation (6.42), while the respective covariance matrix is obtained as follows:

$$\begin{aligned} G_{x_0,y_0,r} &= \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} \Leftrightarrow \\ G_{x_0,y_0,r} &\approx \frac{1}{4} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{x_0}{r} & -\frac{y_0}{r} & \frac{1}{r} \end{bmatrix} [X_{Circ}^T X_{Circ}]^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{x_0}{r} & -\frac{y_0}{r} & \frac{1}{r} \end{bmatrix}^T var(Y_{Circ}) \end{aligned} \quad (6.45)$$

### 6.2.3. Corner Extraction

A corner  $P_j$ , is represented as an invariant point. The corner  $P_j$  is computed through the laser measurements  $R_j$  and  $\theta_j$ , represented in Fig. 6.9, as the following equation:



$$P_j = (R_j + \varepsilon_{Rj}) \begin{bmatrix} \cos(\theta_j + \varepsilon_{\theta}) \\ \sin(\theta_j + \varepsilon_{\theta}) \end{bmatrix} \quad (6.46)$$

Fig. 6.9 Reading parameters of the observation module.

in which the measurement of  $P_j$  is affected with the distance and angle errors:  $\varepsilon_{Rj}$  and  $\varepsilon_{\theta j}$ .

The error of the distance measurement  $\varepsilon_{Rj}$  is modelled with the linear equation:

$$\varepsilon_{Rj} = \varepsilon_k R_j + \varepsilon_c \quad (6.47)$$

in which  $\varepsilon_k$  and  $\varepsilon_c$  is modelled as Gaussian noise with zero mean and standard deviation equal to  $\sigma_k$  and  $\sigma_c$ , respectively. The error  $\varepsilon_\theta$  is modelled as Gaussian noise with zero mean and standard deviation equal to  $\sigma_\theta$ .

The noisy measured point  $P_j$ , can be re-written as the following equations:

$$\begin{aligned} P_j &= (R_j + \varepsilon_k R_j + \varepsilon_c) \begin{bmatrix} \cos(\theta_j + \varepsilon_\theta) \\ \sin(\theta_j + \varepsilon_\theta) \end{bmatrix} \Leftrightarrow \\ P_j &= (R_j + \varepsilon_k R_j + \varepsilon_c) \begin{bmatrix} \cos \theta_j \cos \varepsilon_\theta - \sin \theta_j \sin \varepsilon_\theta \\ \cos \theta_j \sin \varepsilon_\theta + \sin \theta_j \cos \varepsilon_\theta \end{bmatrix} \end{aligned} \quad (6.48)$$

Considering  $\cos(\varepsilon_\theta) \approx 1$ ,  $\sin(\varepsilon_\theta) \approx \varepsilon_\theta$ , the observed noisy point becomes:

$$P_j = (R_j + \varepsilon_k R_j + \varepsilon_c) \begin{bmatrix} \cos \theta_j - \varepsilon_\theta \sin \theta_j \\ \varepsilon_\theta \cos \theta_j + \sin \theta_j \end{bmatrix} \quad (6.49)$$

The estimated observed point,  $\hat{P}_j$  can be computed using the following equation:

$$\hat{P}_j = R_j \begin{bmatrix} \cos \theta_j \\ \sin \theta_j \end{bmatrix} \quad (6.50)$$

Through the equations (6.49) and (6.50), the observed error  $P_j - \hat{P}_j$  can be estimated and given by the expression:

$$P_j - \hat{P}_j = R_j \varepsilon_\theta \begin{bmatrix} -\sin \theta_j \\ \cos \theta_j \end{bmatrix} + (\varepsilon_k R_j + \varepsilon_c) \begin{bmatrix} \cos \theta_j \\ \sin \theta_j \end{bmatrix} \quad (6.51)$$

Using the definition of variance, the variance of  $P_j$ , can be computed as the following equation:

$$\begin{aligned} \text{var}(P_j) &= E[(P_j - \hat{P}_j)(P_j - \hat{P}_j)^T] \Leftrightarrow \\ \text{var}(P_j) &= E[R_j^2 \varepsilon_\theta^2 R_1] + E[(\varepsilon_k R_j + \varepsilon_c)^2 R_2] - E[2R_j \varepsilon_\theta (\varepsilon_k R_j + \varepsilon_c) R_1] \Leftrightarrow \\ \text{var}(P_j) &= R_j^2 E[\varepsilon_\theta^2] R_1 + E[(\varepsilon_k R_j + \varepsilon_c)^2] R_2 - 2R_j E[\varepsilon_\theta (\varepsilon_k R_j + \varepsilon_c)] R_1 \end{aligned} \quad (6.52)$$

in which the matrices  $R_1$ ,  $R_2$  and  $R_3$  are given by the following expressions:

$$R_1 = \begin{bmatrix} -\sin \theta_j \\ \cos \theta_j \end{bmatrix} \begin{bmatrix} -\sin \theta_j \\ \cos \theta_j \end{bmatrix}^T, R_2 = \begin{bmatrix} \cos \theta_j \\ \sin \theta_j \end{bmatrix} \begin{bmatrix} \cos \theta_j \\ \sin \theta_j \end{bmatrix}^T, R_3 = \begin{bmatrix} -\sin \theta_j \\ \cos \theta_j \end{bmatrix} \begin{bmatrix} \cos \theta_j \\ \sin \theta_j \end{bmatrix}^T \quad (6.53)$$

$E[\cdot]$  is the expectation value of  $[\cdot]$ . The expectation values are given by:  $E[\varepsilon_k] = 0$ ,  $E[\varepsilon_c] = 0$ ,  $E[\varepsilon_\theta] = 0$  and  $E[\varepsilon_\theta^2] = \sigma_\theta^2$ ,  $E[\varepsilon_k^2] = \sigma_k^2$  and  $E[\varepsilon_c^2] = \sigma_c^2$ , since  $\varepsilon_\theta$ ,  $\varepsilon_k$  and  $\varepsilon_c$  are errors modeled as Gaussian noise with zero mean and covariances equal to  $\sigma_\theta^2$ ,  $\sigma_k^2$  and  $\sigma_c^2$ ,

respectively. As the errors  $\varepsilon_k$ ,  $\varepsilon_c$  and  $\varepsilon_\theta$  are uncorrelated, the expectation values of  $E[\varepsilon_k \varepsilon_c]$ ,  $E[\varepsilon_\theta \varepsilon_c]$  and  $E[\varepsilon_\theta \varepsilon_k]$  are equal to zero. Therefore, these can be written by the following expressions:

$$\begin{aligned} E[(\varepsilon_k R_j + \varepsilon_c)^2] &= R_j^2 E[\varepsilon_k^2] + 2R_j E[\varepsilon_k \varepsilon_c] + E[\varepsilon_c^2] \Leftrightarrow \\ E[(\varepsilon_k R_j + \varepsilon_c)^2] &= R_j^2 \sigma_k^2 + \sigma_c^2 \end{aligned} \quad (6.54)$$

$$\begin{aligned} E[\varepsilon_\theta (\varepsilon_k R_j + \varepsilon_c)] &= R_j E[\varepsilon_\theta \varepsilon_k] + E[\varepsilon_\theta \varepsilon_c] \Leftrightarrow \\ E[\varepsilon_\theta (\varepsilon_k R_j + \varepsilon_c)] &= 0 \end{aligned} \quad (6.55)$$

Finally, the observed feature  $P_j$  variance is equal to the following expression, trough the equations (6.52), (6.54) and (6.55):

$$\begin{aligned} var(P_j) &= \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} \Leftrightarrow \\ var(P_j) &= R_j^2 \sigma_\theta^2 R_1 + (R_j^2 \sigma_k^2 + \sigma_c^2) R_2 \end{aligned} \quad (6.56)$$

The experiments made in [59] are conducted using a Hokuyo URG LRF, the same used in the work described in this document. Therefore, the standard deviations  $\sigma_k$ ,  $\sigma_c$  and  $\sigma_\theta$ , used in the experiments made in [59], were also used in this PhD work.  $\sigma_k = 4e - 3$  (slope),  $\sigma_c = 1mm$ ,  $\sigma_\theta = 1e - 3$  °.



## 7. Pre-Localization

As in the normal operation of the vehicle, a 3D Matching algorithm is used, aiming to pinpoint its position in the world frame, and, as is described in Chapter of *Localization* 9, the gradients (x and y) and distance matrices are required. To compute these matrices, the occupancy grid is also required, in the three-dimensional space, on the environment.

Therefore, in *pre-localization*, the vehicle needs to be able to localize itself, so that the mapping of the indoor building may be possible. To meet that condition, two possible solutions can be used. They are presented in the following figure, Fig. 7.1. The blocks in dark grey represent the *pre-localization* for both solutions.

The crucial difference between these two solutions is in the initial phase: the first solution is applicable if there is a 2D map of the building (architectural plan), while the second solution, firstly builds the 2D feature map. Between these possibilities, the second was chosen as the final solution, as it will be explained below.

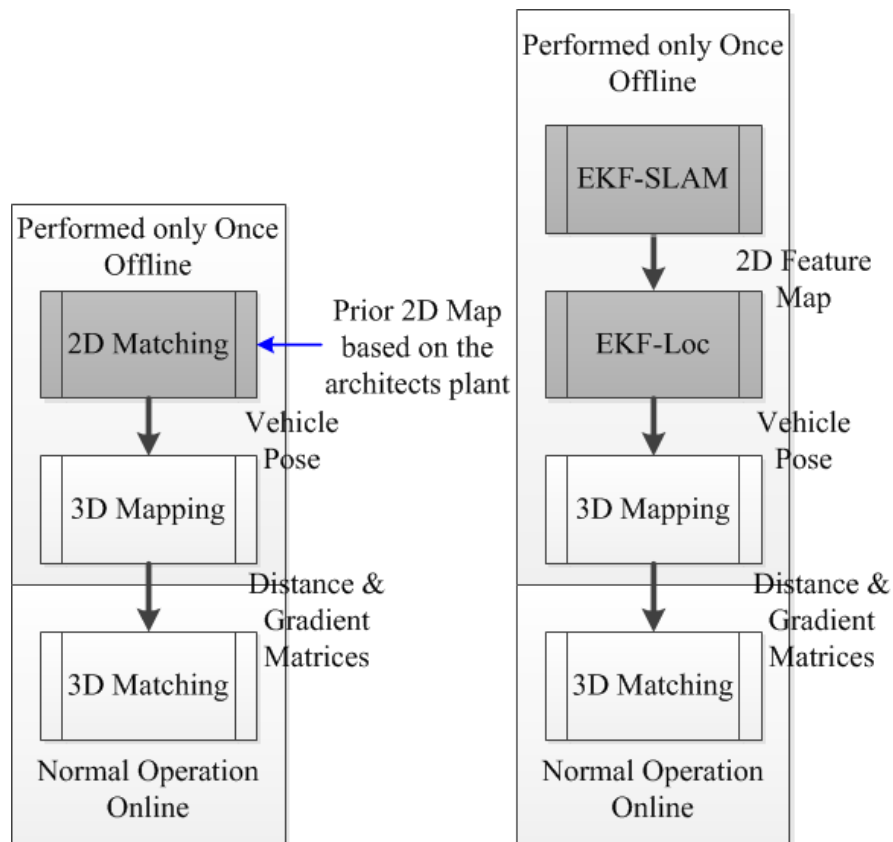


Fig. 7.1 Possible strategies (darker grey blocks) for the *pre-localization* procedure. Left image: *pre-localization* with a 2D Matching algorithm. Right image: *pre-localization* with an EKF-SLAM/EKF-Loc algorithm.

In the first solution, a 2D Matching algorithm can be applied in the 2D space to perform the *pre-localization*. In order to do so, the architect's map of the building can be used to obtain the distance and gradient matrices in x and y directions.

The Fig. 7.1 left shows this strategy. But, in this solution, some drawbacks arise: 1) a preparatory step is necessary to obtain the distance and gradient matrices; 2) the preparation step is performed using the architectural plan of the building, which sometimes is not available, other times it is obsolete and does not contain the real detail of the building, resulting in a bad *pre-localization* behaviour; 3) on the other hand, it can be performed by measuring manually the indoor building, which introduces a source of error.

Therefore, as it is intended, to obtain the building map with high degree of accuracy and without any stage of preparation, a second solution can be considered. This solution uses, during the *pre-localization* procedure, an EKF-SLAM/EKF-Loc algorithm, Fig. 7.1 right.

Without any knowledge on the surrounding environment, the Simultaneous Localization and Mapping (SLAM) allows the vehicle to pinpoint its location inside the world frame and map the surrounding. For these reasons, the final methodology proposed for *pre-localization* is the use of the EKF-SLAM/EKF-Loc procedure.

The EKF-SLAM procedure allows you to obtain, in a first stage, the 2D map on the surrounding environment. This map is a feature map with linear segments and invariant points. In a second stage, still offline, by applying a simpler Extended Kalman Filter with the vehicle state, using the map obtained during the EKF-SLAM stage, it is possible to localize the vehicle (EKF-Loc). The knowledge of the vehicle pose allows applying the next procedure, the *mapping*, Chapter 8.

This redundancy of information (linear segments and invariant points) helps to improve the EKF-SLAM and EKF-Loc accuracy.

Imagine that, during a certain period of time, a *preparation time*, the vehicle can navigate inside the environment which is intended to map, without dynamic objects crossing it.

Initially, with the observation module fixed in the horizontal (tilting LRF in the horizontal), and controlling the vehicle with a joystick, the acquired data in the two-dimensional space can be logged to be then applied during the EKF-SLAM procedure. The acquired points in the two dimensional space constitute the set of points  $P_{2D}$ .

The amount of data  $P_{2D}$  are reduced into linear segments and invariant points, using the *Detection and Extraction* algorithm described in the Chapter 6, thus becoming possible to apply the EKF-SLAM step, as shown in the Fig. 7.2.

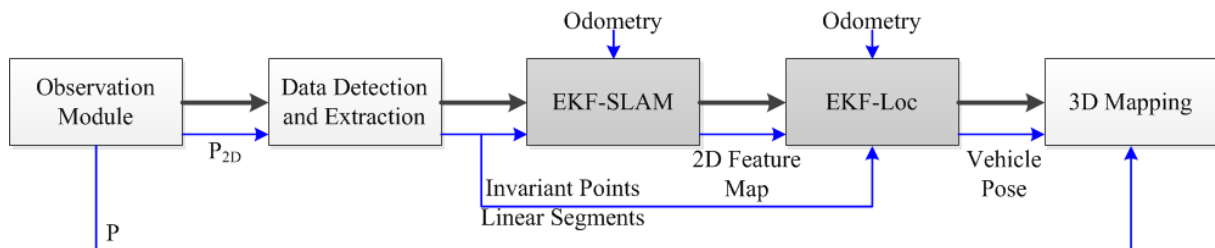


Fig. 7.2 interface between the EKF-SLAM and EKF-Loc procedure and the Data Detection and Extraction Module. At black lines is shown the process sequence, while at blue line is show the data flow.



After, with the *observation module*, by rotating and using the prior knowledge on the 2D feature map, also controlling the vehicle with a joystick, 3D data can be logged to be then applied with the EKF-Loc procedure.

In this last case, the points acquired by the tilting Laser Range Finder are projected in the two-dimensional space. The set of the points  $P$  acquired by the tilting LRF, with coordinates  $(x, y, z)$ , whose  $z$  coordinate is between  $Z_{min}$  and  $Z_{max}$  metres constitute the set of data points  $P_{2D}$ , with coordinates  $(x, y)$ , which are projected in the 2D space. Again, the set of points  $P_{2D}$  are transformed into linear segments and invariant points, using also the *Detection and Extraction* algorithm, allowing the application of the EKF Loc algorithm. The vehicle position obtained in the EKF-Loc step and the set of points  $P$  with coordinates  $(x, y, z)$  are used during the *mapping* procedure.

The values of the parameters  $Z_{min}$  and  $Z_{max}$  are presented in the Chapter of results, Chapter 10.

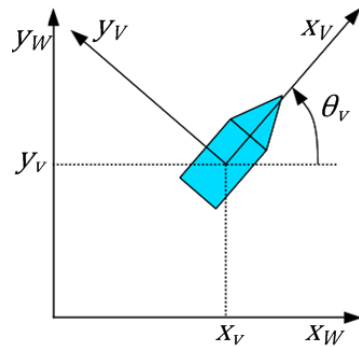
### 7.1. Filter's State

The nomenclature and state representation of the EKF-SLAM and EKF-Loc used in this Chapter shall be introduced in this sub-section.

The Fig. 7.3 shows the world frame, with axis  $(x_W, y_W)$  and the vehicle frame  $(x_V, y_V)$ . The vehicle pose is represented by the vector  $(x_v, y_v, \theta_v)$ . The vehicle referential is in the middle between the two traction wheels of the robot, as described in Chapter 5. Therefore, the origin of the vehicle frame has coordinates, in relation to the world frame, equal to the vehicle 2D position  $(x_v, y_v)$ , and rotation, also in relation to the world frame, equal to the angle  $\theta_v$ .

When the Extended Kalman Filter is applied to a SLAM problem, the filter's vector state is not fixed. In this case, the vector state is composed of the vehicle state  $X_v$  and by the map state  $X_m^W$ , both represented in the world frame. Therefore, the vector state of the filter, in the world frame, is equal to the following:

$$X = \begin{bmatrix} X_v \\ X_m^W \end{bmatrix} \quad (7.1)$$



$$X_v = \begin{bmatrix} x_v \\ y_v \\ \theta_v \end{bmatrix} \quad (7.2)$$

Fig. 7.3 Absolute referential  $(x_W, y_W)$ . Vehicle's referential  $(x_v, y_v)$ . Vehicle state  $X_v$ .

The map vector state  $X_m^W$  is increased every time that a new feature is obtained. Since the features are walls, corners and columns, represented by linear segments and invariant points, the map state appears:

$$X_m^W = \begin{bmatrix} X_{Lin}^W \\ X_{Pnt}^W \end{bmatrix} \quad (7.3)$$

in which  $X_{Lin}^W$  is the vector state corresponding to the linear segments found in the environment.  $X_{Pnt}^W$  is the vector state corresponding to the set of invariant points (corners and columns) in the navigation area. Therefore, the map state  $X_m^W$  can be represented by the following vector:

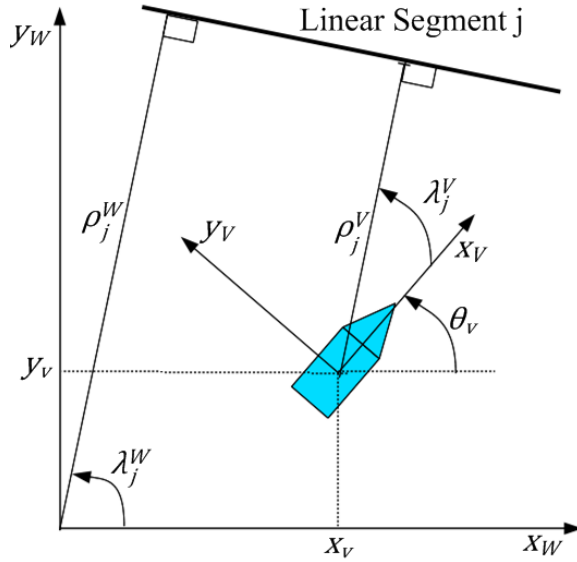
$$X_m^W = [X_{Lin1}^W \quad \dots \quad X_{LinM}^W \quad X_{Pnt1}^W \quad \dots \quad X_{PntN}^W]^T \quad (7.4)$$

in which  $M$  is the number of linear segment features.  $N$  is the number of invariant points.

As can be seen in the Fig. 7.4, the generic linear segment  $j$  in the vehicle referential is represented with parameters  $\rho_j^V$  and  $\lambda_j^V$ , by the vector:

$$X_{Linj}^V = \begin{bmatrix} \rho_j^V \\ \lambda_j^V \end{bmatrix} \quad (7.5)$$

A linear segment  $j$  is kept in the state vector as shown in the following equation. The state  $X_{Linj}^W$  is the polar representation of a line, in which  $\rho_j^W$  is the perpendicular distance of the linear segment in relation to the world's referential origin. Parameter  $\lambda_j^W$  is the angle between the absolute referential axis  $x_W$  and the perpendicular with the linear segment. See Fig. 7.4.



$$X_{Linj}^W = \begin{bmatrix} \rho_j^W \\ \lambda_j^W \end{bmatrix} \quad (7.6)$$

Fig. 7.4 Absolute referential  $(x_W, y_W)$ . Vehicle's referential  $(x_V, y_V)$ . Vehicle state  $X_v$ . Linear segment representing a wall, with state  $X_{Linj}^W$ .

A corner or column  $j$  is represented in the vehicle referential, with Cartesian coordinates  $(x_{Pntj}^V, y_{Pntj}^V)$ . The observed state appears equal to:

$$X_{Pntj}^V = \begin{bmatrix} x_{Pntj}^V \\ y_{Pntj}^V \end{bmatrix} \quad (7.7)$$

The corner or a column  $j$  represented as a point in world frame, is expressed in the feature map vector  $X_m^W$ , as is shown in equation (7.8), with parameters  $x_{pnt}^W$  and  $y_{pnt}^W$ , in relation to the origin of the world referential as is shown in Fig. 7.5 and Fig. 7.6.

$$X_{Pntj}^W = \begin{bmatrix} x_{Pntj}^W \\ y_{Pntj}^W \end{bmatrix} \quad (7.8)$$

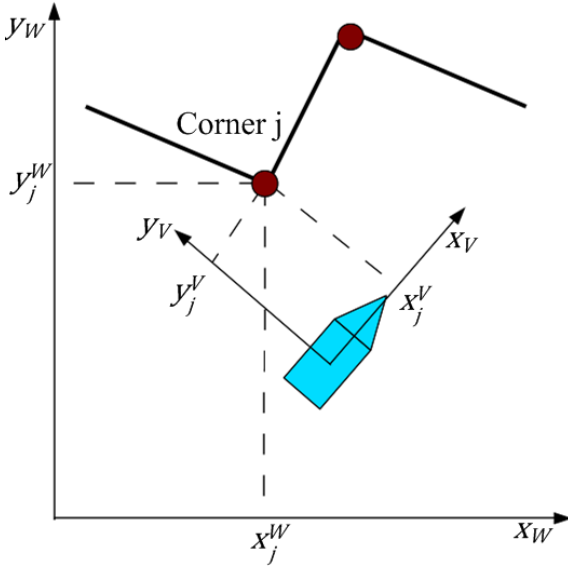


Fig. 7.5 Point  $j$  representing a corner, with state  $X_{Pntj}^W$ .

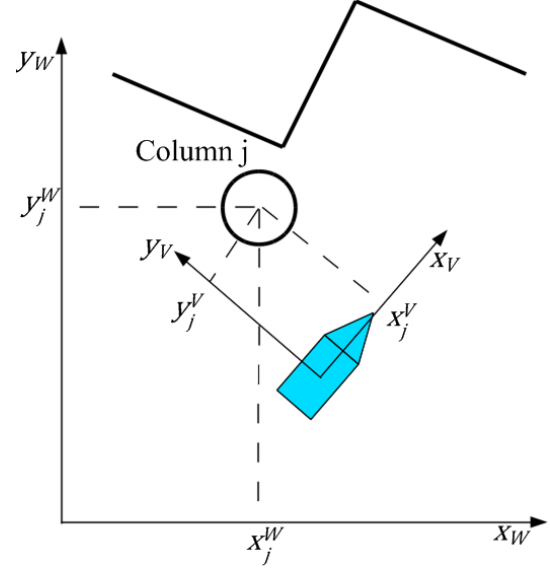


Fig. 7.6 Point  $j$  representing a column, with state  $X_{Pntj}^W$ .

Furthermore the equation (7.1), of the filter's vector state, can be re-written as follows:

$$X = \begin{bmatrix} \begin{bmatrix} x_v \\ y_v \\ \theta_v \end{bmatrix}^T & \begin{bmatrix} \rho_1^W \\ \lambda_1^W \end{bmatrix}^T & \dots & \begin{bmatrix} \rho_M^W \\ \lambda_M^W \end{bmatrix}^T & \begin{bmatrix} x_{Pnt1}^W \\ y_{Pnt1}^W \end{bmatrix}^T & \dots & \begin{bmatrix} x_{PntN}^W \\ y_{PntN}^W \end{bmatrix}^T \end{bmatrix}^T \quad (7.9)$$

The estimation of the Filter's state is represented by vector  $\hat{X}$ , composed of the following components:

$$\hat{X} = \begin{bmatrix} \hat{X}_v \\ \hat{X}_m^W \end{bmatrix} \quad (7.10)$$

in which, the estimation of the vehicle state  $\hat{X}_v$  is set up by the variables  $(\hat{x}_v, \hat{y}_v, \hat{\theta}_v)$  and the estimative of the map vector state  $\hat{X}_m^W$ , is represented by the following vector:

$$\hat{X}_m^W = [\hat{X}_{Lin1}^W \quad \dots \quad \hat{X}_{LinM}^W \quad \hat{X}_{Pnt1}^W \quad \dots \quad \hat{X}_{PntN}^W]^T \quad (7.11)$$

The estimation of a linear segment  $j$  in the vehicle and world frames is represented by the following vectors, respectively:

$$\hat{X}_{Linj}^V = \begin{bmatrix} \hat{\rho}_j^V \\ \hat{\lambda}_j^V \end{bmatrix}, \quad \hat{X}_{Linj}^W = \begin{bmatrix} \hat{\rho}_j^W \\ \hat{\lambda}_j^W \end{bmatrix} \quad (7.12)$$

While the estimation of a point  $j$  in the vehicle and world frames is represented by the vectors:

$$\hat{X}_{Pntj}^V = \begin{bmatrix} \hat{x}_{Pntj}^V \\ \hat{y}_{Pntj}^V \end{bmatrix}, \quad \hat{X}_{Pntj}^W = \begin{bmatrix} \hat{x}_{Pntj}^W \\ \hat{y}_{Pntj}^W \end{bmatrix} \quad (7.13)$$

A new feature is firstly stored in the list of unmapped features (the list of unmapped linear segments and the list of unmapped invariant points), until being observed a minimum number of iterations and consequently added in the map state vector ( $\hat{X}_m^W$ ). This is performed to avoid or reduce the introduction of wrong features in the map vector.

The minimum number of iterations required to introduce a new feature in the vehicle state is indicated bellow, sub-section 7.4, with the respective justification.

Both lists are represented in the world frame. The list of unmapped linear segments is represented by the following vector:

$$X_{UnMpLin}^W = [X_{UnMpLin1}^W \quad \dots \quad X_{UnMpLinE}^W]^T \quad (7.14)$$

in which  $E$  represents the number of unmapped linear segments in the list and the state vector  $X_{UnMpLinj}^W$  is represented as the following:

$$X_{UnMpLinj}^W = \begin{bmatrix} \rho_{UnMpLinj}^W \\ \lambda_{UnMpLinj}^W \end{bmatrix} \quad (7.15)$$

The list of invariant points is represented by the following vector:

$$X_{UnMpPnt}^W = [X_{UnMpPnt1}^W \quad \dots \quad X_{UnMpPntF}^W]^T \quad (7.16)$$

in which  $F$  represents the number of unmapped invariant points in the list, and the state vector  $X_{UnMpPntj}^W$  is represented as follows:

$$X_{UnMpPntj}^W = \begin{bmatrix} x_{UnMpPntj}^W \\ y_{UnMpPntj}^W \end{bmatrix} \quad (7.17)$$

Finally, an acquired linear segment, through the Data Detection and Extraction Module, described in Chapter 6, is called observed linear segment and is represented by the following vectors, in the vehicle and world frames:

$$X_{ObsLinj}^V = \begin{bmatrix} \rho_{Obsj}^V \\ \lambda_{Obsj}^V \end{bmatrix}, \quad X_{ObsLinj}^W = \begin{bmatrix} \rho_{Obsj}^W \\ \lambda_{Obsj}^W \end{bmatrix} \quad (7.18)$$

Also, the acquired invariant points, called observed points, also obtained with the Data Detection and Extraction Module, are represented in the vehicle and world frames as follows:

$$X_{ObsPntj}^V = \begin{bmatrix} x_{ObsPntj}^V \\ y_{ObsPntj}^V \end{bmatrix}, \quad X_{Pntj}^W = \begin{bmatrix} x_{ObsPntj}^W \\ y_{ObsPntj}^W \end{bmatrix} \quad (7.19)$$

In the following sub-sections, it will be necessary to compute the estimation of linear segments in the vehicle frame  $\hat{X}_{Linj}^V$ , through the estimation of the same feature but, in coordinates of the world frame  $\hat{X}_{Linj}^W$ . The same occurs in the estimation of invariant points. Also, in the case of the observed features, they are obtained in the vehicle referential  $X_{ObsLinj}^V$  and  $X_{ObsPntj}^V$ , and required to be transformed in the world referential:  $X_{ObsLinj}^W$  and  $X_{ObsPntj}^W$ .

Therefore, these transformations will be explained in this introductive sub-section, to be used during the following sub-sections.

In the case of the linear segment, there are two possible transformations that need to be distinguished when transforming the linear segment parameters from the vehicle to the world frame and vice-versa. These two situations depend on whether the vehicle is inside or outside of the inner zone defined by the linear segment  $X_{Linj}^W$ , represented in the figures in grey, Fig. 7.7 and Fig. 7.8.

Let us now define the situation when the vehicle is inside the inner zone, shown in Fig. 7.7, it will be called as *inner* situation. On the contrary, when the vehicle is outside of the inner zone defined by  $X_{Linj}^W$ , as shown in Fig. 7.8, it will be called as *outer* situation.

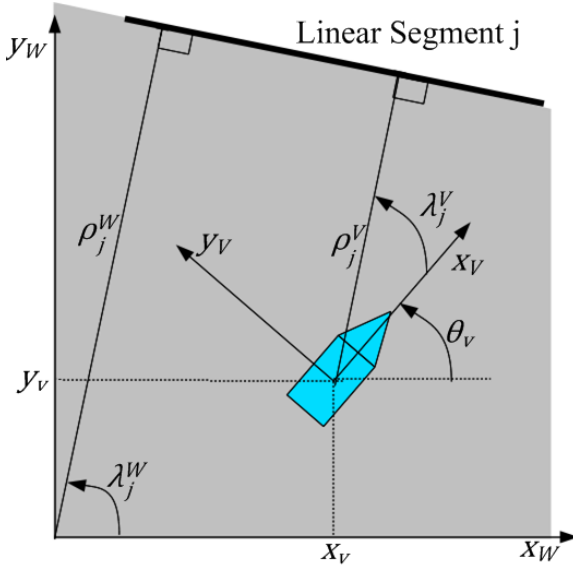


Fig. 7.7 Example of an inner situation. The inner zone is represented is grey.

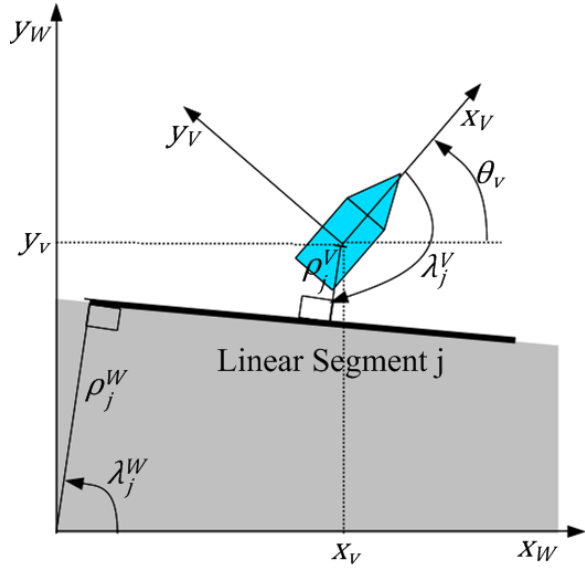


Fig. 7.8 Example of a outer situation. The inner zone is represented is grey.

In the inner situation, the linear segment  $j$  written in the world referential coordinates can be transformed in the vehicle referential, with the following equation:

$$X_{Linj}^V = \begin{bmatrix} \rho_j^W - x_v \cos \lambda_j^W - y_v \sin \lambda_j^W \\ \lambda_j^W - \theta_v \end{bmatrix} \quad (7.20)$$

In the outer situation, the previous equation need to be re-written as follows:

$$X_{Linj}^V = \begin{bmatrix} -\rho_j^W + x_v \cos \lambda_j^W + y_v \sin \lambda_j^W \\ \lambda_j^W - \theta_v + \pi \end{bmatrix} \quad (7.21)$$

Therefore, this transformation can be generalized to both cases, through the expression:

$$X_{Linj}^V = \begin{bmatrix} sign \cdot (\rho_j^W - x_v \cos \lambda_j^W - y_v \sin \lambda_j^W) \\ \lambda_j^W - \theta_v + (1 - sign) \cdot \frac{\pi}{2} \end{bmatrix} \quad (7.22)$$

in which *sign* is equal to -1 in the inner situation and 1 on the opposite situation.

The linear segment *j*, written in the vehicle referential, can also be transformed in the world referential. In the inner situation, the transformation is performed with the following equation:

$$X_{Linj}^W = \begin{bmatrix} \rho_j^V + x_v \cos(\lambda_j^V + \theta_v) + y_v \sin(\lambda_j^V + \theta_v) \\ \lambda_j^V + \theta_v \end{bmatrix} \quad (7.23)$$

In the outer situation, the previous equation is written as follows:

$$X_{Linj}^W = \begin{bmatrix} -\rho_j^V + x_v \cos(\lambda_j^V + \theta_v - \pi) + y_v \sin(\lambda_j^V + \theta_v - \pi) \\ \lambda_j^V + \theta_v - \pi \end{bmatrix} \quad (7.24)$$

Generalizing to both the cases, the transformation is equal to:

$$X_{Linj}^W = \begin{bmatrix} sign \cdot \rho_j^V + x_v \cos(\lambda_j^V + \theta_v + (sign - 1) \cdot \frac{\pi}{2}) + y_v \sin(\lambda_j^V + \theta_v + (sign - 1) \cdot \frac{\pi}{2}) \\ \lambda_j^V + \theta_v + (sign - 1) \cdot \frac{\pi}{2} \end{bmatrix} \quad (7.25)$$

Still about the linear segments, it is necessary to distinguish the inner and outer situations. Imagine that it is necessary to compute the linear segment, written in coordinates of the world frame, in coordinates of the vehicle frame. The equation of the linear segment  $X_{Linj}^W$  can be expressed as follows:

$$\rho_j^W = x^W \cos \lambda_j^W + y^W \sin \lambda_j^W \quad (7.26)$$

in which  $x^W$  and  $y^W$  are the coordinates of a generic point of the linear segment represented in the world frame. Therefore, the linear segment  $X_{Linj}^W$  corresponds to an inner situation, as shown in Fig. 7.7, if the following condition is met. When this condition is not met, the linear segment  $X_{Linj}^W$ , corresponds to an outer situation, as shown in Fig. 7.8.

$$\rho_j^W \leq x_v \cos \lambda_j^W + y_v \sin \lambda_j^W \quad (7.27)$$

Imagine now that we intend to write in coordinates of the world referential, the linear segment  $X_{Linj}^V$ , expressed in the coordinates of the vehicle. In this case, both situations, the inner and outer, are computed using equations (7.23) and (7.24). Afterwards, two solutions will be obtained, but only one will be considered the correct one. If the solution obtained with

equation (7.23) meets the condition (7.27), it will be the correct solution. On the contrary, if the solution of (7.24) does not meet the condition (7.27), it will be the correct solution.

In the case of the invariant point, the transformation between its representation in the world frame and its representation in the vehicle frame, is expressed with the equation:

$$X_{Pntj}^V = \begin{bmatrix} +(x_{Pntj}^W - x_v) \cos \theta_v + (y_{Pntj}^W - y_v) \sin \theta_v \\ -(x_{Pntj}^W - x_v) \sin \theta_v + (y_{Pntj}^W - y_v) \cos \theta_v \end{bmatrix} \quad (7.28)$$

On the contrary, the representation of a point in the world referential through its coordinates in the vehicle referential, can be expressed as follows:

$$X_{Pntj}^W = \begin{bmatrix} x_{Pntj}^V \cos \theta_v - y_{Pntj}^V \sin \theta_v + x_v \\ x_{Pntj}^V \sin \theta_v + y_{Pntj}^V \cos \theta_v + y_v \end{bmatrix} \quad (7.29)$$

## 7.2. EKF-SLAM Procedure Cycle

In this sub-section will be detailed presented the EKF-SLAM cycle. Pseudo-code is used in this sub-section as way to describe the EKF-SLAM cycle, as shown in Algorithm 7.1.

The EKF-SLAM cycle, is composed of four crucial steps: 1) the Kalman Filter Prediction stage; 2) the Association Module; 3) the Kalman Filter Update stage; and finally 4) the Delete Repeated Features Module.

In the first step, the previous Filter's estimated state  $\hat{X}(k|k)$  and respective covariance  $P(k|k)$ , are propagated using the odometry inputs.

After, the association module compute the correspondences between the actual observed list of linear segments and invariant points,  $LList^V$  and  $PList^V$ , and the features estimated on the Filter's vector state  $\hat{X}(k+1|k)$ . These associations are returned in the vectors  $H_{Lin}$  and  $H_{Pnt}$ .

The observed features with no association in the Filter's vector state  $\hat{X}(k+1|k)$  have a corresponding index in the association vectors  $H_{Lin}$  or  $H_{Pnt}$  equal to -1. For those observed features, it is tried the association with the list of unmapped features, already defined as  $X_{UnMpLin}^W$  and  $X_{UnMpPnt}^W$ . These correspondences are returned in the association vectors  $UnH_{Lin}$  and  $UnH_{Pnt}$ . Again, the observed features with no association on the list of unmapped features have a corresponding index equal to -1, in the association vectors,  $UnH_{Lin}$  or  $UnH_{Pnt}$ .

These vectors ( $H_{Lin}$ ,  $H_{Pnt}$ ,  $UnH_{Lin}$  and  $UnH_{Pnt}$ ), are used in the following phase, the Kalman Filter Update stage.

In the third stage, Kalman Filter Update, the state and covariance propagated by the odometry inputs  $\hat{X}(k+1|k)$  and  $P(k+1|k)$ , is updated and the new Filter's state and respective covariance is computed,  $\hat{X}(k+1|k+1)$  and  $P(k+1|k+1)$ . To do that, the observation and its estimation are computed with the knowledge of correspondences, in the vectors  $H_{Lin}$  and  $H_{Pnt}$ .

The entire set of features in the list of unmapped features, have a score value, which indicates, the number of times that each of those unmapped feature appeared. Therefore, the score is increased, for the features in the list of unmapped features, with an association in the vectors  $UnH_{Lin}$  and  $UnH_{Pnt}$ .

The features with no association in the vectors  $H_{Lin}$ ,  $H_{Pnt}$ ,  $UnH_{Lin}$  and  $UnH_{Pnt}$  are introduced in the list of unmapped features, with a score equal to 1.

Still about the Kalman Filter Update stage (third step of the EKF-SLAM), the features in the list of unmapped features with a score higher than a minimum value are added in Filter's vector state  $\hat{X}(k+1|k)$  and deleted from the list of unmapped features.

Finally, during the fourth step, in intervals of *delRepFeatCycle* iterations, the repeated features in the estimated state vector  $\hat{X}(k+1|k+1)$  are deleted. In the following subsections this four stages will be explained in more detail.

Algorithm 7.1 EKF-SLAM cycle.

<b><i>Cycle_SLAM(Odo, PList<sup>V</sup>, SList<sup>V</sup>)</i></b>
<b>loop</b>  <b>EKF – SLAM Prediction</b>  $\hat{X}(k+1 k), P(k+1 k) \leftarrow \text{Predict}(\text{Odo}, \hat{X}(k k), P(k k))$  <b>Association Module</b>  $H_{Lin}, H_{Pnt}, UnH_{Lin}, UnH_{Pnt} \leftarrow \text{VectorOfAssociations}(LList^V, PList^V, \hat{X}(k+1 k))$  <b>EKF – SLAM Update</b>  $[Z] \leftarrow \text{Observation}(LList^V, PList^V)$  $[\hat{h}] \leftarrow \text{Estimated Observation}(H_{Lin}, H_{Pnt}, \hat{X}(k+1 k))$  $\hat{X}(k+1 k+1), P(k+1 k+1) \leftarrow \text{Update}(Z, \hat{h}, \hat{X}(k+1 k), P(k+1 k))$  $\hat{X}(k+1 k+1), P(k+1 k+1) \leftarrow \text{AddNewFeatures}(Z, \hat{X}(k+1 k+1), P(k+1 k+1))$  <b>Delete Repeated Features</b>  <b>do</b> (in cycles of <i>delRepFeatCycle</i> iterations)  $\text{Delete Repeated Points}(\hat{X}(k+1 k+1), P(k+1 k+1))$  $\text{Delete Repetead Lines}(\hat{X}(k+1 k+1), P(k+1 k+1))$  <b>end do</b>  <b>end loop</b>

Along this Chapter the following nomenclature shall be used: the matrix  $0^{(a) \times (b)}$  represents the zero matrix with dimension  $(a) \times (b)$ . The matrix  $I^{(a) \times (b)}$  represents the identity matrix with dimension  $(a) \times (b)$ . The matrix  $A^{(a) \times (b)}$  represents a generic matrix A, with dimension  $(a) \times (b)$ .



### 7.2.1. Kalman Filter Prediction Stage

Every cycle, the odometry model predicts, by using odometry data ( $Odo$ ), the previous vector state  $\hat{X}(k|k)$  and covariance  $P(k|k)$ , a new vector state  $\hat{X}(k+1|k)$  and covariance matrix  $P(k+1|k)$ .

In the prediction stage, only the vehicle state ( $x_v, y_v, \theta_v$ ), will be changed with odometry data, since the features, represented in the world referential will remain static.

Aiming to model the vehicle's kinematic model, the following transition function was used, considering the state  $X(k|k)$ :

$$f(X(k|k), Odo, q) = \begin{bmatrix} f_v(X_v(k|k), Odo, q_v) \\ X_{Lin1}(k|k) \\ \dots \\ X_{LinM}(k|k) \\ X_{Pnt1}(k|k) \\ \dots \\ X_{PntN}(k|k) \end{bmatrix} \quad (7.30)$$

in which the vehicle kinematic model is given by the following expression, [62]:

$$f_v(X_v(k|k), Odo, q_v) = \begin{bmatrix} x_v(k|k) + d \cdot \cos\left(\theta_v(k|k) + \frac{\Delta\theta}{2}\right) + \varepsilon_{xv} \\ y_v(k|k) + d \cdot \sin\left(\theta_v(k|k) + \frac{\Delta\theta}{2}\right) + \varepsilon_{yv} \\ \theta_v(k|k) + \Delta\theta + \varepsilon_\theta \end{bmatrix} \quad (7.31)$$

The kinematic model's error  $q_v$  was modelled as additive white noise, with Gaussian distribution, with an expected value equal to zero ( $\hat{q}_v = 0$ ) and covariance  $Q_v$ . The white noise  $q_v$  is constituted by the error associated with the vehicle state:

$$q_v = [\varepsilon_{xv} \quad \varepsilon_{yv} \quad \varepsilon_{\theta v}]^T \quad (7.32)$$

The estimation of the Filter's state after the Kalman Filter Prediction stage is obtained using the estimation of the previous state  $\hat{X}(k|k)$  and odometry data as shown in the following expression:

$$\hat{X}(k+1|k) = f(\hat{X}(k|k), Odo) \Leftrightarrow \begin{bmatrix} \hat{x}_v(k|k) + d \cdot \cos\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \\ \hat{y}_v(k|k) + d \cdot \sin\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \\ \hat{\theta}_v(k|k) + \Delta\theta \\ \hat{X}_{Lin1}(k|k) \\ \dots \\ \hat{X}_{LinM}(k|k) \\ \hat{X}_{Pnt1}(k|k) \\ \dots \\ \hat{X}_{PntN}(k|k) \end{bmatrix} \quad (7.33)$$

To compute the prediction covariance it is necessary to use the following equation, as it can be seen in more detail in the book [53]:

$$P(k+1|k) = \frac{\partial f}{\partial X} P(k|k) \frac{\partial f^T}{\partial X} + \frac{\partial f}{\partial q_v} Q_v \frac{\partial f^T}{\partial q_v} \quad (7.34)$$

The covariance  $Q_v$  increases with the estimated distance and rotation of the vehicle ( $d$  and  $\Delta\theta$ ):

$$Q_v = \begin{bmatrix} \left[ d \cdot \cos\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \cdot \sigma_d \right]^2 & 0 & 0 \\ 0 & \left[ d \cdot \sin\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \cdot \sigma_d \right]^2 & 0 \\ 0 & 0 & (d \cdot \sigma_{d\theta})^2 + (\Delta\theta \cdot \sigma_\theta)^2 \end{bmatrix} \quad (7.35)$$

in which the standard deviations  $\sigma_d$ ,  $\sigma_\theta$  and  $\sigma_{d\theta}$  were determined as described in Chapter 5.

When the vehicle moves 1 metre in front the estimated error on the odometry translational and rotational displacement is  $\sigma_d$  and  $\sigma_{d\theta}$ , respectively. When the vehicle rotates 1 radian, the estimated odometry error on the vehicle rotation is equal to  $\sigma_\theta$ .

Therefore, when the vehicle showed no movement, the kinematic model error, represented by  $Q_v$ , is zero. In the other hand, the variance of the states  $\hat{x}_v$  and  $\hat{y}_v$  increase with the quadratic displacement of the vehicle in the x and y axis, respectively. The variance of  $\hat{\theta}_v$  increases with the quadratic translational and rotational displacement, as it can be seen in equation (7.35).

The gradient of the transition model in order to the odometry error  $q_v$  is equal to the following matrix:

$$\begin{aligned} \nabla f_{q_v} &= \frac{\partial f}{\partial q_v} \Leftrightarrow \\ \nabla f_{q_v} &= \begin{bmatrix} I^{(3) \times (3)} & 0^{(3) \times (2M+2N)} \\ 0^{(2M+2N) \times (3)} & 0^{(2M+2N) \times (2M+2N)} \end{bmatrix} \end{aligned} \quad (7.36)$$

The gradient of the transition model in order to the Filter's state can be obtained with the following expression:

$$\begin{aligned} \nabla f_X(\hat{X}(k|k), Odo) &= \frac{\partial f}{\partial X} \Leftrightarrow \\ \nabla f_X(\hat{X}(k|k), Odo) &= \begin{bmatrix} \nabla f_{X_v}(\hat{X}_v(k|k), Odo) & 0^{(3) \times (2M+2N)} \\ 0^{(2M+2N) \times (3)} & I^{(2M+2N) \times (2M+2N)} \end{bmatrix} \end{aligned} \quad (7.37)$$

in which the gradient of the vehicle kinematic model  $f_v$ , in order to the vehicle state  $X_v$  is obtained with the following matrix:

$$\nabla f_{X_v}(\hat{X}_v(k|k), Odo) = \begin{bmatrix} 1 & 0 & -d \cdot \sin\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \\ 0 & 1 & d \cdot \cos\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \\ 0 & 0 & 1 \end{bmatrix} \quad (7.38)$$

### 7.2.2. Association Module

The Data Detection and Extraction Module obtains a list of points  $PList^V$  and linear segments  $LList^V$ , relative to the vehicle referential. The linear segment list can be defined as the following expression:

$$LList^V = [X_{ObsLin1}^V \quad \dots \quad X_{ObsLinG}^V] \quad (7.39)$$

in which  $X_{ObsLinj}^V$  is equal to the equation (7.18) and  $G$  is the number of observed linear segments. The point list can be defined as the following expression:

$$PList^V = [X_{ObsPnt1}^V \quad \dots \quad X_{ObsPntLH}^V] \quad (7.40)$$

in which  $X_{ObsPntj}^V$  is equal to the equation (7.19) and  $H$  is the number of observed points (corners or columns).

Both these lists, in the association module, need to be computed in the world frame, using the equations already described in sub-section 7.1. Therefore, the observed linear segment list, expressed in the world frame, can be represented by the following vector:

$$LList^W = [X_{ObsLin1}^W \quad \dots \quad X_{ObsLinG}^W] \quad (7.41)$$

The observed invariant points list, represented in the world referential, can be expressed as follows:

$$PList^W = [X_{ObsPnt1}^W \quad \dots \quad X_{ObsPntH}^W] \quad (7.42)$$

Given a point list and a linear segment list in the world referential, the vehicle state  $\hat{X}_v(k+1|k)$  and the map state  $\hat{X}_m^W(k+1|k)$  can be updated by comparing the actual observations and the state stored in the Filter's vector. Also, new features can be added to the Filter's vector. To do so it is necessary to perform the association between those observed features and the estimated state vector or the list of unmapped features, as shown in the image of Fig. 7.9.

Firstly, we test whether each observed feature ( $X_{ObsLinj}^W$  or  $X_{ObsPntj}^W$ ) from the lists  $LList^W$  and  $LPnt^W$ , has a correspondence in the map state vector  $\hat{X}_m^W$ . These correspondences are returned in the vectors  $H_{Lin}$  and  $H_{Pnt}$ .

For the observed features with no association in  $\hat{X}_m^W$ , i.e. with an index in  $H_{Lin}$  or  $H_{Pnt}$  equal to -1, depending if it is a linear segment or a point case, respectively, it is tested for associations with the list of unmapped features,  $X_{UnMpLin}^W$  or  $X_{UnMpPnt}^W$ . In that case, the correspondences are returned in the vectors  $UnH_{Lin}$  and  $UnH_{Pnt}$ .

Both these association modules, represented in Fig. 7.9, are constituted with two crucial phases: 1) the test of the individual compatibility, which using a set of "gates", associate to each observed feature an corresponding candidate in the state vector or in the unmapped list

of features, respectively; 2) from the first step, can result observed features with the same associated candidate. To prevent the association of the same candidate to different observed features, the test of the joint compatibility is performed.

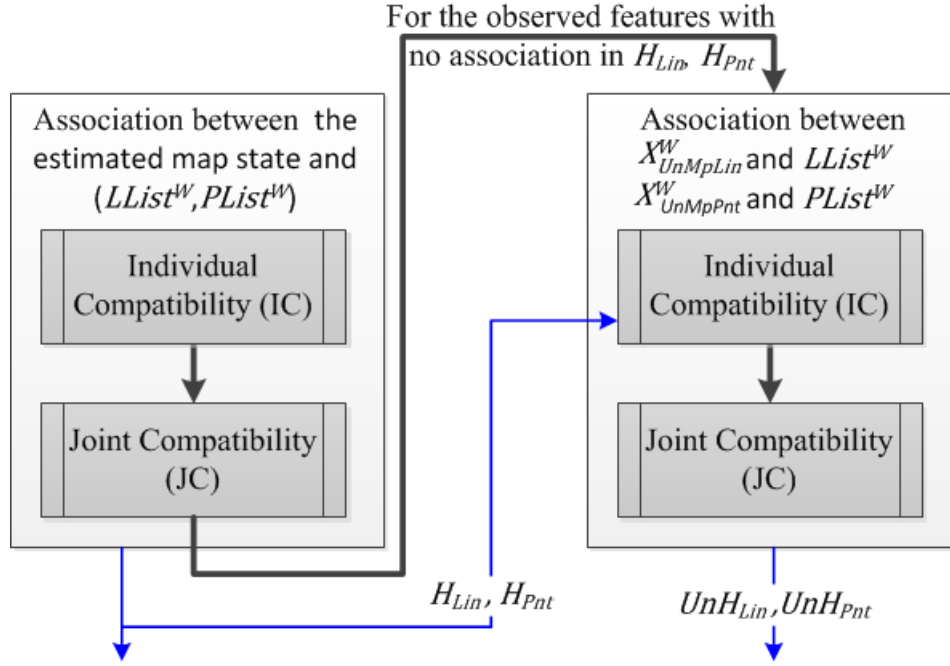


Fig. 7.9 Association modules. Correspondence between the observed features and the estimated map state  $\hat{X}_m^W$ .  
Correspondence between the observed features and the list of unmapped features.

Trying with all the feature points in the map state vector, the pair  $(X_{ObsPnti}^W, \hat{X}_{Pntj}^W)$  is individual compatible only if the following conditions are met:

- 1) The Euclidian distance between  $X_{ObsPnti}^W$  and  $\hat{X}_{Pntj}^W$  is lower than the adjustable parameter  $\delta_{d_{Pnt}}$ .
- 2) The fusion value  $\omega_{Pnt}$  between  $X_{ObsPnti}^W$  and  $\hat{X}_{Pntj}^W$  is the smallest between the pairs that meet the condition 1), where  $\omega_{Pnt}$  is given by the expression:

$$\omega_{Pnt} = \sqrt{\left(\frac{\hat{x}_{Pntj}^W - x_{ObsPnti}^W}{\sigma_{x_{Pntj}}}\right)^2 + \left(\frac{y_{Pntj}^W - y_{ObsPnti}^W}{\sigma_{y_{Pntj}}}\right)^2} \quad (7.43)$$

The standard deviations  $\sigma_{x_{Pntj}}$  and  $\sigma_{y_{Pntj}}$  are the diagonal covariances, corresponding to the mapped invariant point  $\hat{X}_{Pntj}^W$ , in the covariance matrix  $P$ .

After the application of the individual compatibility, each observed feature can be assigned with a mapped feature candidate, or with no association. The joint compatibility is tested, thus verifying if the same candidate is attributed in different observed features. In the positive cases, the better observed feature wins the candidate, while the others are assigned with no association. An observed feature wins the candidate if its fusion value  $\omega_{Pnt}$  is the lower.

Finally, the observed points assigned with no association are tested again to find pairs in the list of unmapped points  $X_{UnMpPntj}^W$ . To do that, it is performed the test of the individual

compatibility. Consider again the limit  $\delta_{d_{pnt}}$ , the pair  $(X_{ObsPnti}^W, X_{UnMpPntj}^W)$  is associated only if the conditions are met:

- 1) The Euclidian distance between  $X_{ObsPnti}^W$  and  $X_{UnMpPntj}^W$  is lower than a limit  $\delta_{d_{pnt}}$ .
- 2) The previous computed Euclidian distance is the smallest between the pairs that meet the condition 1).

After applied the individual compatibility, again, the joint compatibility is applied for the list of unmapped points  $X_{UnMpPnt}^W$ .

We shall now consider the observed linear segment  $X_{ObsLini}^W$  and the mapped feature  $\hat{X}_{Linj}^W$  represented in the following figure, Fig. 7.10. Consider also, the projection of the observed linear segment in the mapped feature, equal to  $(X_{ObsLini}^W)_{proj}$ .

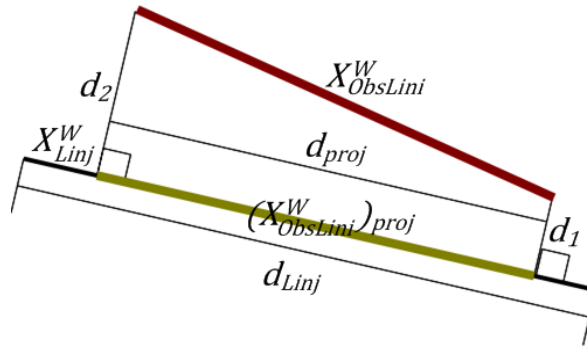


Fig. 7.10  $X_{ObsLini}^W$  in red.  $\hat{X}_{Linj}^W$  in black. Projection of  $X_{ObsLini}^W$  in green.

The size of the feature  $\hat{X}_{Linj}^W$  is equal to  $d_{Linj}$ , while the size of the projected observed feature is equal to  $d_{proj}$ . The overlapping rate value  $O_r$ , between  $X_{ObsLini}^W$  and  $\hat{X}_{Linj}^W$  is computed in the following way:

$$O_r = \frac{d_{Linj} + d_{proj}}{\max(d_{Linj}, d_{proj})} \quad (7.44)$$

The overlapping value will be equal to 2 when there is complete overlap. The value of  $O_r$  is equal to 1 when there is no overlap.

The disparity value  $d_{\perp Lin}$  represents the distance between  $X_{ObsLini}^W$  and  $\hat{X}_{Linj}^W$ . Consider the distance  $d_1$  and  $d_2$ , representing the distances between the first and last points of the observed linear segment  $X_{ObsLini}^W$  and its projection in  $\hat{X}_{Linj}^W$ . The disparity value is computed as follows:

$$d_{\perp Lin} = \frac{d_1 + d_2}{2} \quad (7.45)$$

Considering the adjustable values  $\delta_{\perp Lin}$  and  $\delta_{\lambda Lin}$ , the pair  $(X_{ObsLini}^W, \hat{X}_{Linj}^W)$  is individual compatible only if the conditions are met:

- 1) If the disparity value  $d_{\perp Lin}$ , between the estimated linear segment  $\hat{X}_{Linj}^W$  and the observed is lower or equal than  $\delta_{\perp Lin}$ .
- 2) If the angle distance,  $d_{\lambda Lin}$ , between the angle  $\lambda_j^W$  of  $\hat{X}_{Linj}^W$  and the observed linear segment  $X_{ObsLini}^W$  is lower or equal than  $\delta_{\lambda Lin}$ .

3) If there is a minimum overlap rate  $O_r$  equal to  $\min O_r$ , between the linear segments  $\hat{X}_{Linj}^W$  and  $X_{ObsLini}^W$ .

4) The fusion value  $\omega_{Lin}$ , between  $\hat{X}_{Linj}^W$  and  $X_{ObsLini}^W$  is the smallest between the pairs that meet conditions 1), 2) and 3). In which  $\omega_{Lin}$  is determined with the following expression:

$$\omega_{seg} = \frac{\frac{d_{\perp Lin} + d_{\lambda Lin}}{\sigma_{\rho Linj} + \sigma_{\lambda Linj}} + \frac{1}{O_r}}{\frac{1}{\sigma_{\rho Linj}} + \frac{1}{\sigma_{\lambda Linj}}} \quad (7.46)$$

Thus, a mapped linear segment  $\hat{X}_{Linj}^W$  with a higher uncertainty in its parameters, i.e. higher  $\sigma_{\rho Linj}$  and  $\sigma_{\lambda Linj}$ , has a lower fusion value to be paired with  $X_{ObsLini}^W$ .

Such as it occurs with the point feature association, after the individual compatibility is applied, the different linear segments observed can be associated with the same mapped feature. Therefore, the joint compatibility is applied. When different observed linear segments have the same candidate, the one with a lower fusion value  $\omega_{Lin}$ , wins the candidate, while the others became with no association.

Again, as it occurs with the association described for the invariant points, the observed linear segments assigned with no association are tested again to find pairs in the list of unmapped linear segments  $X_{UnMpLin}^W$ . The association between an observed linear segment, with the state  $X_{UnMpLinj}^W$ , has similarities with the association described above. Therefore, the pair  $(X_{ObsLini}^W, X_{UnMpLinj}^W)$  is individual compatible if the conditions are met:

1) If the disparity value  $d_{\perp Lin}$ , between the unmapped linear segment  $X_{UnMpLinj}^W$  and the observed is lower or equal than  $\delta_{\perp Lin}$ .

2) If the angle distance,  $d_{\lambda Lin}$ , between the angle  $\lambda_{UnMpLj}^W$  of  $X_{UnMpLinj}^W$  and the observed linear segment  $X_{ObsLini}^W$  is lower or equal than  $\delta_{\lambda Lin}$ .

3) If there are a minimum overlap rate  $O_r$ ,  $\min O_r$ , between the linear segments  $X_{UnMpLinj}^W$  and  $X_{ObsLini}^W$ .

4) The fusion value,  $\omega_{Lin}$ , between  $X_{UnMpLinj}^W$  and  $X_{ObsLini}^W$  is the smallest between the pairs that meet conditions 1), 2) and 3). The fusion value  $\omega_{Lin}$  is determined with the following expression:

$$\omega_{Lin} = d_{\perp Lin} + d_{\lambda Lin} + \frac{1}{O_r} \quad (7.47)$$

Also after the individual compatibility is applied, in the association between the observed linear segments and the unmapped, the joint compatibility is tested.

### 7.2.3. Kalman Filter Update Stage

We shall now consider that the estimated state vector has M linear segments and N invariant points. The dimension of the state vector, enter in account with the estimation of the vehicle state  $\hat{X}_v$ , is equal to  $(3+2N+2M)$ , since the vehicle state has three associated variables and each features has two associated variables.

In this sub-section there will be an explanation on how the entire state vector is updated using the associated observed features to the map state vector  $\hat{X}_m^W(k+1|k)$ . Also it is explained how the features are introduced in the state vector.

Through the test of compatibility explained in the previous sub-section, the observed features are associated with the ones that already exist in the estimated map vector  $\hat{X}_m^W$ . We shall now consider the vectors  $H_{Lin}$  and  $H_{Pnt}$ , obtained after the application of the association module. Through these vectors, it is possible to define the association pairs, between the estimated features and the observed:  $(\hat{X}_{Linj}^W, X_{ObsLinj}^V)$  and  $(\hat{X}_{Pntj}^W, X_{ObsPntj}^V)$ . Consider also that the number of associations between observed linear segments and the estimated is G. The number of associations between the observed invariant points and the estimated is H.

In the case of a linear segment, the observation model, is equal to  $X_{Linj}^V$  in the polar form, written in the vehicle coordinates:

$$h_{Linj} = \begin{bmatrix} \rho_j^V \\ \lambda_j^V \end{bmatrix} \quad (7.48)$$

The estimated observation, in the case of the linear segment is equal to:

$$\begin{aligned} \hat{h}_{Linj} &= \begin{bmatrix} \hat{\rho}_j^V \\ \hat{\lambda}_j^V \end{bmatrix} (k+1|k) \Leftrightarrow \\ \hat{h}_{Linj} &= \hat{X}_{Linj}^V(k+1|k) \end{aligned} \quad (7.49)$$

By using the equation (7.22), the estimated observation in the case of the linear segment j, can be written in the following way:

$$\hat{h}_{Linj} = \begin{bmatrix} sign \cdot [\hat{\rho}_j^W - \hat{x}_v \cos \hat{\lambda}_j^W - \hat{y}_v \sin \hat{\lambda}_j^W] \\ \hat{\lambda}_j^W - \hat{\theta}_v + (1 - sign) \cdot \frac{\pi}{2} \end{bmatrix} (k+1|k) \quad (7.50)$$

The observation of a linear segment can be written as the following expression:

$$\begin{aligned} Z_{Linj} &= h_{Linj} + r_{Linj} \Leftrightarrow \\ Z_{Linj} &= \begin{bmatrix} \rho_j^V + \varepsilon_{\rho j} \\ \lambda_j^V + \varepsilon_{\lambda j} \end{bmatrix} \end{aligned} \quad (7.51)$$

in which  $r_{Linj}$  is the error associated with the observed feature  $X_{ObsLinj}^V$ , modelled as Gaussian noise with zero mean ( $\hat{r}_{Linj} = 0$ ) and covariance matrix  $R_{Linj}$ . The error  $r_{Linj}$  is represented by the vector:

$$r_{Linj} = \begin{bmatrix} \varepsilon_{\rho j} \\ \varepsilon_{\lambda j} \end{bmatrix} \quad (7.52)$$

The observation of a linear segment can be written in another way, as function of the observed feature:

$$Z_{Linj} = X_{ObsLinj}^V \quad (7.53)$$

The associated innovation to the estimated linear segment  $j$ , is equal to the following equation:

$$\begin{aligned} V_{Linj} &= Z_{Linj} - \hat{h}_{Linj} \Leftrightarrow \\ V_{Linj} &= X_{ObsLinj}^V - \hat{X}_{Linj}^V \end{aligned} \quad (7.54)$$

The gradient of the observation module  $h_{Linj}$  in order to the observation model noise  $r_{Linj}$  is equal to the identity matrix, while the gradient in order to the state vector, equivalent to the mapped linear segment  $j$ , is equal to  $\nabla h_{Linj}^X$  and is given by the equations:

$$\nabla h_{Linj}^X = \begin{bmatrix} \nabla h_{Linj}^{x_v} & \nabla h_{Linj}^{x_m^W} \end{bmatrix} \quad (7.55)$$

$$\nabla h_{Linj}^{x_v} = \begin{bmatrix} -sign \cdot \cos \hat{\lambda}_j^W & -sign \cdot \sin \hat{\lambda}_j^W & 0 \\ 0 & 0 & -1 \end{bmatrix} (k+1|k) \quad (7.56)$$

$$\nabla h_{Linj}^{x_m^W} = \begin{bmatrix} 0^{(2) \times (2j-2)} & \nabla h_{Linj}^{x_m^W} & 0^{(2) \times (2M+2N-2j)} \end{bmatrix} \quad (7.57)$$

$$\nabla h_{Linj}^{x_m^W} = \begin{bmatrix} sign & -sign \cdot [\hat{y}_c \cos \hat{\theta}_v - \hat{x}_v \sin \hat{\theta}_v] \\ 0 & 1 \end{bmatrix} (k+1|k) \quad (7.58)$$

The matrix of the covariance associated to the observation model's error, is given by the following expression:

$$R_{Linj} = \begin{bmatrix} K_{\sigma_\rho} + \sigma_{\rho_j}^2 & \sigma_{\rho\lambda_j} \\ \sigma_{\rho\lambda_j} & K_{\sigma_\lambda} + \sigma_{\lambda_j}^2 \end{bmatrix} \quad (7.59)$$

in which  $K_{\sigma_\rho}$  and  $K_{\sigma_\lambda}$  are adjustable parameters whose values are given with the respective justification at the sub-section 7.4. The standard deviations  $\sigma_{\rho_j}^2$ ,  $\sigma_{\lambda_j}^2$  and  $\sigma_{\rho\lambda_j}$  are estimated for the observed linear segment  $j$ , as explained in the Data Detection and Extraction Module, with the equation (6.36). Therefore, the final innovation vector, of the linear segment features, considering all the observed linear segments, appears equal to:

$$V_{Lin} = [V_{Lin1} \quad \dots \quad V_{Lin2} \quad \dots \quad V_{LinG}]^T \quad (7.60)$$

The final gradient vector, in order to the Filter's state, can be written as the following expression:

$$\nabla h_{Lin}^X = [\nabla h_{Lin1}^X \quad \dots \quad \nabla h_{Linj}^X \quad \dots \quad \nabla h_{LinG}^X]^T \quad (7.61)$$

Finally, the resultant covariance matrix associated with the list of the observed features can be written as follows:



$$R_{Lin} = \begin{bmatrix} R_{Lin1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & R_{LinG} \end{bmatrix} \quad (7.62)$$

We shall now consider the case of the invariant point feature. The observation model of an invariant point  $j$  is equal to  $X_{Pntj}^V$ :

$$h_{Pntj} = \begin{bmatrix} x_{Pntj}^V \\ y_{Pntj}^V \end{bmatrix} \quad (7.63)$$

The estimation of the observation module corresponding to the mapped feature  $j$  is given by:

$$\begin{aligned} \hat{h}_{Pntj} &= \begin{bmatrix} x_{Pntj}^V \\ y_{Pntj}^V \end{bmatrix} (k+1|k) \Leftrightarrow \\ \hat{h}_{Pntj} &= \hat{X}_{Pntj}^V (k+1|k) \end{aligned} \quad (7.64)$$

Using the equation (7.28), the estimated observation in the case of an invariant point  $j$ , can be written in the following way:

$$l_x = \hat{x}_{Pntj}^W (k+1|k) - \hat{x}_v (k+1|k) \quad (7.65)$$

$$l_y = \hat{y}_{Pntj}^W (k+1|k) - \hat{y}_v (k+1|k) \quad (7.66)$$

$$\hat{h}_{Pntj} = \begin{bmatrix} l_x \cos \hat{\theta}_v + l_y \sin \hat{\theta}_v \\ l_y \cos \hat{\theta}_v - l_x \sin \hat{\theta}_v \end{bmatrix} (k+1|k) \quad (7.67)$$

The observation of an invariant point can be written as the following expression:

$$\begin{aligned} Z_{Pntj} &= h_{Pntj} + r_{Pntj} \Leftrightarrow \\ Z_{Pntj} &= \begin{bmatrix} x_{Pntj}^V + \varepsilon_{xj} \\ y_{Pntj}^V + \varepsilon_{yj} \end{bmatrix} \end{aligned} \quad (7.68)$$

in which the error  $r_{Pntj}$  is the error associated with the observed feature  $X_{ObsPntj}^V$ , modelled as Gaussian noise with zero mean ( $\hat{r}_{Pntj} = 0$ ) and covariance matrix  $R_{Pntj}$ . The error  $r_{Pntj}$  is represented by the vector:

$$r_{Pntj} = \begin{bmatrix} \varepsilon_{xj} \\ \varepsilon_{yj} \end{bmatrix} \quad (7.69)$$

The observation of the invariant point can be written in another way, depending on the observed point  $j$ :

$$Z_{Pntj} = X_{ObsPntj}^V \quad (7.70)$$

The innovation associated with the estimated invariant point  $j$  is equal to the following equation:

$$\begin{aligned} V_{Pntj} &= Z_{Pntj} - \hat{h}_{Pntj} \Leftrightarrow \\ V_{Pntj} &= X_{ObsPntj}^V - \hat{X}_{Pntj}^V \end{aligned} \quad (7.71)$$

The gradient of the observation module  $h_{Pntj}$ , in order to the noise of the observed feature equal to  $r_{Pntj}$ , is the identity matrix, while the gradient in order to the state vector, equivalent to the estimated point  $j$ , is equal to  $\nabla h_{Pntj}^X$  and is given by the equations:

$$\nabla h_{Pntj}^X = \begin{bmatrix} \nabla h_{Pntj}^{X_v} & \nabla h_{Pntj}^{X_m^W} \end{bmatrix} \quad (7.72)$$

$$\nabla h_{Pntj}^{X_v} = \begin{bmatrix} -\cos \hat{\theta}_v & -\sin \hat{\theta}_v & -l_x \sin \hat{\theta}_v + l_y \cos \hat{\theta}_v \\ +\sin \hat{\theta}_v & -\cos \hat{\theta}_v & -l_x \cos \hat{\theta}_v - l_y \sin \hat{\theta}_v \end{bmatrix} (k+1|k) \quad (7.73)$$

$$\nabla h_{Pntj}^{X_m^W} = \begin{bmatrix} 0^{(2) \times (2j-2)} & \nabla h_{Pnt}^{X_{Pntj}^W} & 0^{(2) \times (2N+2K-2j)} \end{bmatrix} \quad (7.74)$$

$$\nabla h_{Pnt}^{X_{Pntj}^W} = \begin{bmatrix} \cos \hat{\theta}_v & \sin \hat{\theta}_v \\ -\sin \hat{\theta}_v & \cos \hat{\theta}_v \end{bmatrix} (k+1|k) \quad (7.75)$$

The matrix of the covariance associated to the observation model's error, is given by the following expression:

$$R_{Pntj} = \begin{bmatrix} K_{\sigma_x} + \sigma_{x_j}^2 & \sigma_{xy_j} \\ \sigma_{xy_j} & K_{\sigma_y} + \sigma_{y_j}^2 \end{bmatrix} \quad (7.76)$$

in which  $K_{\sigma_x}$  and  $K_{\sigma_y}$  are adjustable parameters which value is presented and justified in the sub-section 7.4. The values  $\sigma_{x_j}^2$ ,  $\sigma_{y_j}^2$  and  $\sigma_{xy_j}$  are the covariances estimated for the observed invariant point  $j$ , as explained in the Data Detection and Extraction Module, through the equations (6.45) or (6.56), depending if the observed invariant point is corresponding to a column or a corner, respectively.

Therefore, the final innovation vector, of the invariant point features, considering the associated observed invariant points, appears equal to:

$$V_{Pnt} = [V_{Pnt1} \quad \dots \quad V_{Pntj} \quad \dots \quad V_{PntH}]^T \quad (7.77)$$

The final gradient vector in order to the Filter's state can be written as the following expression:

$$\nabla h_{Pnt}^X = [\nabla h_{Pnt1}^X \quad \dots \quad \nabla h_{Pntj}^X \quad \dots \quad \nabla h_{PntH}^X]^T \quad (7.78)$$

Finally, the resultant covariance matrix, associated with the list of the observed features can be written as follows:

$$R_{Pnt} = \begin{bmatrix} R_{Pnt1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & R_{PntH} \end{bmatrix} \quad (7.79)$$

The total innovation can be obtained with the vector:

$$V(k+1) = [V_{Lin} \quad V_{Pnt}]^T \quad (7.80)$$

The total gradient in order to the noise of the observed features is the identity matrix, while the total gradient in order to the Filter's state is given by the following equation:

$$\nabla h_X = [\nabla h_{Lin}^X \quad \nabla h_{Pnt}^X]^T \quad (7.81)$$

The resultant covariance of the entire set of observed features is given by the following matrix:

$$R = \begin{bmatrix} R_{Lin} & 0^{(G) \times (H)} \\ 0^{(H) \times (G)} & R_{Pnt} \end{bmatrix} \quad (7.82)$$

The Kalman Filter equations during the Update stage can be seen in more detail in [53]. In that way, the new vector state  $X(k+1|k+1)$ , using the Extended Kalman Filter's equations in the Update stage, appears equal to:

$$\hat{X}(k+1|k+1) = \hat{X}(k+1|k) + W(k+1) V(k+1) \quad (7.83)$$

in which  $W(k+1)$  is the Kalman Filter's Gain equal to:

$$W(k+1) = P(k+1|k) \nabla h_X^T [\nabla h_X P(k+1|k) \nabla h_X^T + R]^{-1} \quad (7.84)$$

Finally, the new covariance vector will appear equal to:

$$P(k+1|k+1) = P(k+1|k) - W(k+1) \nabla h_X P(k+1|k) \quad (7.85)$$

Now, also in the Kalman Filter Update stage, there will be an introduction of the new features and the respective covariance in the Filter's map vector  $\hat{X}_m^W(k+1|k+1)$ , and in the covariance matrix  $P(k+1|k+1)$ , respectively.

The association model also returns the vectors  $UnH_{Lin}$  and  $UnH_{Pnt}$ , which for the observed features with no association in  $\hat{X}_m^W$ , have the correspondence in relation to the list of unmapped features. Through these vectors it is possible to obtain the associated pairs, between the unmapped features and the observed:  $(X_{UnMpLinj}^W, X_{ObsLinj}^V)$  and  $(X_{UnMpPntj}^W, X_{ObsPntj}^V)$ . These vectors will be -1 for the observed features with no association in the list of unmapped features.

In the case of an association, the score of the corresponding unmapped feature  $X_{UnMpLinj}^W$  or  $X_{UnMpPntj}^W$ , is increased. If the score of the associated feature is higher than a value *MinScore*, it is added to the map state vector  $\hat{X}_m^W$  and deleted from the list of unmapped features.

If there are not a correspondence, then, the observation  $X_{ObsLinj}^W$  or  $X_{ObsPntj}^W$  is added to the list of unmapped features, as a potential new feature.

Now, will be explained how the introduction of new features is done in the Extended Kalman Filter. A linear segment as function of the parameters of an observed linear segment  $X_{ObsLinj}^W$ , is represented by the following expressions, deduced in equation (7.25):

$$\lambda_{Obsj}^W = (\lambda_{Obsj}^V + \varepsilon_{\lambda j}) + \theta_v(k+1|k+1) + (sign - 1) \cdot \frac{\pi}{2} \quad (7.86)$$

$$X_{Linj}^W = \begin{bmatrix} sign \cdot (\rho_{Obsj}^V + \varepsilon_{\rho j}) + x_v \cos \lambda_{Obsj}^W + y_v \sin \lambda_{Obsj}^W \\ \lambda_{Obsj}^W \end{bmatrix} (k+1|k+1) \quad (7.87)$$

in which the noise  $r_{Linj}$  is setup by the random noise  $\varepsilon_{\lambda j}$  and  $\varepsilon_{\rho j}$ , modelled as Gaussian noise with expected value  $\hat{r}_{Linj}$ , equal to zero and covariance  $R_{Linj}$ , defined at equation (7.59).

The estimation of a linear segment as function of an observed linear segment  $X_{ObsLinj}^W$  is computed as the following equation:

$$\hat{X}_{Linj}^W = X_{ObsLinj}^W \Leftrightarrow \quad (7.88)$$

$$\hat{\lambda}_{Obsj}^W = \lambda_{Obsj}^V + \hat{\theta}_v(k+1|k+1) + (sign - 1) \cdot \frac{\pi}{2} \quad (7.89)$$

$$X_{ObsLinj}^W = \begin{bmatrix} sign \cdot \rho_{Obsj}^V + \hat{x}_v \cos \hat{\lambda}_{Obsj}^W + \hat{y}_v \sin \hat{\lambda}_{Obsj}^W \\ \hat{\lambda}_{Obsj}^W \end{bmatrix} (k+1|k+1) \quad (7.90)$$

Therefore, when the observed linear segment  $j$  is added to the filter's state vector, because its score on the list of unmapped linear segments is higher than a value *MinScore*, the new estimation of the state comes:

$$\hat{X}(k+1|k+1) = \begin{bmatrix} \hat{X}_v^T & \hat{X}_{Lin1}^W & \dots & \hat{X}_{LinM}^W & X_{ObsLinj}^W & \hat{X}_{Pnt1}^W & \dots & \hat{X}_{PntN}^W \end{bmatrix}^T \quad (7.91)$$

The new covariance matrix appears equal to:

$$P(k+1|k+1) = \nabla h_{Lin}^X P(k+1|k+1) \nabla h_{Lin}^{X^T} + \nabla h_{Lin}^{r_{Linj}} R_{Linj} \nabla h_{Lin}^{r_{Linj}T} \quad (7.92)$$

The gradient of the estimation of a linear segment, in order to the state  $X$ , is equal to  $\nabla h_{Linj}^X$  and can be computed with the expressions:

$$\nabla h_{Lin}^X = \begin{bmatrix} \nabla h_{Linj}^{Xv} & \mathbf{0}_{(3) \times (2M+2N+2)} \\ \mathbf{0}_{(2M+2N+2) \times (3)} & \mathbf{0}_{(2M+2N+2) \times (2M+2N+2)} \end{bmatrix} \quad (7.93)$$

$$\nabla h_{Linj}^{Xv} = \begin{bmatrix} \cos \hat{\lambda}_{Obsj}^W & \sin \hat{\lambda}_{Obsj}^W & -\hat{x}_v \sin \hat{\lambda}_{Obsj}^W + \hat{y}_v \cos \hat{\lambda}_{Obsj}^W \\ 0 & 0 & 1 \end{bmatrix} (k+1|k+1) \quad (7.94)$$

The gradient of the estimation of a linear segment, in order to the white noise  $r_{Linj}$ , is given by the following equations:

$$\nabla h_{Lin}^{r_{Linj}} = \begin{bmatrix} 0^{(3+2M+2N) \times (2)} \\ \nabla h_{Linj}^{r_{Linj}} \end{bmatrix} \quad (7.95)$$

$$\nabla h_{Linj}^{r_{Linj}} = \begin{bmatrix} signal & -\hat{x}_v \sin \hat{\lambda}_{Obsj}^W + \hat{y}_v \cos \hat{\lambda}_{Obsj}^W \\ 0 & 1 \end{bmatrix} (k+1|k+1) \quad (7.96)$$

In the case of an observed point, an invariant point, as function of the parameters of the observed invariant point  $X_{ObsPntj}^W$ , can be written as follows. To do that the equation already deduced (7.29) is used:

$$X_{Pntj}^W = \begin{bmatrix} (x_{ObsPntj}^V + \varepsilon_{xj}) \cos \theta_v - (y_{ObsPntj}^V + \varepsilon_{yj}) \sin \theta_v + x_v \\ (x_{ObsPntj}^V + \varepsilon_{xj}) \sin \theta_v + (y_{ObsPntj}^V + \varepsilon_{yj}) \cos \theta_v + y_v \end{bmatrix} (k+1|k+1) \quad (7.97)$$

in which the noise  $r_{Pntj}$  is setup by the random noise  $\varepsilon_{xj}$  and  $\varepsilon_{yj}$ , modelled as Gaussian noise with expected value  $\hat{r}_{Pntj}$ , equal to zero and covariance  $R_{Pntj}$ , defined at equation (7.59).

The estimation of an invariant point as function of an observed  $X_{ObsPntj}^W$  is equal to:

$$\hat{X}_{Pntj}^W = X_{ObsPntj}^W \Leftrightarrow \hat{X}_{Pntj}^W = \begin{bmatrix} x_{ObsPntj}^V \cos \hat{\theta}_v - y_{ObsPntj}^V \sin \hat{\theta}_v + \hat{x}_v \\ x_{ObsPntj}^V \sin \hat{\theta}_v + y_{ObsPntj}^V \cos \hat{\theta}_v + \hat{y}_v \end{bmatrix} (k+1|k+1) \quad (7.98)$$

Then, when it is added to the Filter's state vector, the new estimative of the state comes:

$$\hat{X}(k+1|k+1) = \begin{bmatrix} \hat{X}_v^T & X_{Lin1}^{W^T} & \dots & X_{LinM}^{W^T} & X_{Pnt1}^{W^T} & \dots & X_{PntN}^{W^T} & X_{ObsPntj}^{W^T} \end{bmatrix}^T \quad (7.99)$$

Therefore, the new estimative of the covariance matrix is given by the expression:

$$P(k+1|k+1) = \nabla h_{Pnt}^X P(k+1|k+1) \nabla h_{Pnt}^{X^T} + \nabla h_{Pnt}^{r_{Pntj}} R_{Pntj} \nabla h_{Pntj}^{r_{Pntj}^T} \quad (7.100)$$

The gradient of the observation module in order to the state  $X$  comes equal to:

$$\nabla h_{Pnt}^X = \begin{bmatrix} \nabla h_{Pntj}^{X_v} & 0^{(3) \times (2M+2N+2)} \\ 0^{(2M+2N+2) \times (3)} & 0^{(2M+2N+2) \times (2M+2N+2)} \end{bmatrix} \quad (7.101)$$

$$\nabla h_{Pntj}^{X_v} = \begin{bmatrix} 1 & 0 & -x_{ObsPntj}^V \sin \hat{\theta}_v - y_{Pntj}^V \cos \hat{\theta}_v \\ 0 & 1 & +x_{ObsPntj}^V \cos \hat{\theta}_v - y_{Pntj}^V \sin \hat{\theta}_v \end{bmatrix} (k+1|k+1) \quad (7.102)$$

The gradient of the observation model in order to the noise  $r_{pnt}$ , is given by the following equations:

$$\nabla h_{Pnt}^{r_{Pntj}} = \begin{bmatrix} 0^{(3+2M+2N) \times (2)} \\ \nabla h_{Pntj}^{r_{Pntj}} \end{bmatrix} \quad (7.103)$$

$$\nabla h_{Pntj}^{r_{Pntj}} = \begin{bmatrix} \cos \hat{\theta}_v & -\sin \hat{\theta}_v \\ \sin \hat{\theta}_v & \cos \hat{\theta}_v \end{bmatrix} (k+1|k+1) \quad (7.104)$$

### 7.2.4. Delete Repeated Features Module

The filter's vector state can have different features whose state is equal or similar. These features are repeated and one of them should be deleted.

Periodically, in intervals of *delRepFeatCycle* iterations (see Algorithm 7.1), the vector state is inspected aiming to find pairs of repeated features  $(\hat{X}_{Pnti}^W, \hat{X}_{Pntj}^W)$  or  $(\hat{X}_{Lini}^W, \hat{X}_{Linj}^W)$ , merging the repeated ones.

To verify the association between features the individual compatibility need to be verified. The individual compatibility, used in this case is based on the works [41] and [58]. It is determined using the Mahalanobis distance.

If the Mahalanobis distance is lower than  $\alpha^2$ , the features  $(\hat{X}_{Pnti}^W, \hat{X}_{Pntj}^W)$  or  $(\hat{X}_{Lini}^W, \hat{X}_{Linj}^W)$  should be considered a pair. The value assumed to  $\alpha$  was 0.95 representing an association with an confidence level of 95%, [58].

The Mahalanobis distance between the feature i and feature j is computed using the following equations, where *Feat* represents the linear segment or invariant point case:

$$mahalanobis = V_{Feat} S_{vv}^{-1} V_{Feat}^T \quad (7.105)$$

$$V_{Feat} = \hat{X}_{Feati}^W - \hat{X}_{Featj}^W \quad (7.106)$$

$$S_{vv} = P(\hat{X}_{Feati}^W) + P(\hat{X}_{Featj}^W) \quad (7.107)$$

in which  $P(\hat{X}_{Feati}^W)$  and  $P(\hat{X}_{Featj}^W)$  are the corresponding covariance matrices of the features in the Filter's covariance matrix *P*. These features are the pairs  $(\hat{X}_{Lini}^W, \hat{X}_{Linj}^W)$  or  $(\hat{X}_{Pnti}^W, \hat{X}_{Pntj}^W)$  in the case of a linear segment or invariant point, respectively.

Finally, the pair  $(\hat{X}_{Lini}^W, \hat{X}_{Linj}^W)$  will be associated and merged in only one linear segment, if there is an overlap between the two linear segments and the Mahalanobis distance meets the following equation. In the case of the pair of invariant points  $(\hat{X}_{Pnti}^W, \hat{X}_{Pntj}^W)$ , they are merged in only one feature, if the following equation is met.

$$mahalanobis < 0.95^2 \quad (7.108)$$

### 7.3. EKF-Loc Cycle

As already said, after the EKF-SLAM procedure described at the previous sub-section, it is applied, still offline, the EKF-Loc cycle, aiming the vehicle accurate localization using the 2D feature map, previously computed. The crucial difference between the EKF-SLAM cycle and the EKF-Loc cycle is the fact that in the last the state vector is only constituted by the vehicle state:

$$X = [x_v \quad y_v \quad \theta_v]^T \quad (7.109)$$

Hence, in the EKF-Loc cycle the crucial goal is, using the features and the respective covariances, of the map previously built during the EKF-SLAM cycle, only estimate the vehicle state  $\hat{X}_v$ . Therefore, contrary to the EKF-SLAM cycle, during the EKF-Loc cycle: 1)

the map previously built, is not updated; 2) new features are not added in the state vector; and 3) features are not deleted from the map.

All the matrices are computed as shown in the previous sub-section, with exceptions for the gradient matrices  $\nabla h_{Linj}^X$  and  $\nabla h_{Pntj}^X$ , which are equal to  $\nabla h_{Linj}^{X_v}$  and  $\nabla h_{Pntj}^{X_v}$ , computed as shown in the equations (7.56) and (7.73), respectively.

#### 7.4. Pre-Localization Parameters

The Table 7.1 presents the parameters of the update step in the EKF-SLAM and EKF-Loc cycles used during the *pre-localization*, while the Table 7.2 shows the adjustable parameters of the association module

The algorithm was tested with different values of  $K_{\sigma_\rho}$ ,  $K_{\sigma_\lambda}$ ,  $K_{\sigma_x}$  and  $K_{\sigma_y}$  in the gap of  $[0.04, 0.3]$ , in intervals of 0.01. The best performance was achieved for the values shown in the Table 7.1. The best performance was found, adjusting these parameters making up the *pre-localization* smother as possible.

We shall now consider that the vehicle moves at a speed of 0.2 m/s, during the *pre-localization* procedure. With a time cycle of 100 milliseconds, the vehicle will move a maximum distance  $maxDist$ , between time cycles, equal to  $0.2 \times 0.1 = 0.02 \text{ metres}$ . Smother as possible means a localization without a variation of the vehicle position higher than three times the value of  $maxDist$ , which is equal to  $0.06 \text{ metres}$ .

The parameters  $\delta_{\perp Lin}$ ,  $\delta_{\lambda Lin}$  and  $O_r$  are related with the association between linear segment features. As they are three validation gates, the values of Table 7.2 have a high degree of security. In indoor environments the walls, doors and furniture is disposed in rectangle angles, in the parallel and orthogonal.  $\delta_{\lambda Lin}$  is used to associate linear segments which are parallel, therefore it is also used to despise the association between orthogonal linear segments. Therefore the value of  $90^\circ/2 = 45^\circ$  for the parameter  $\delta_{\lambda Lin}$  is a logical value.

On the other hand the parameter  $\delta_{\perp Lin}$ , is used to despise the association between distant parallel linear segments, as is example the walls of a corridor. The distance between the walls of a corridor, where the vehicle passes, is at least 0.5 metres, the diameter of the vehicle's footprint. Therefore the value of  $\delta_{\perp Lin}$  is 0.5 metres.

It is considered that the overlap between two linear segments, allowing they association, need to be at least 0.25 metres. Only linear segments with at least 0.7 metres are extracted from the raw data set. Therefore, it is only allowed the association between the observed linear segment  $X_{ObsLinj}^W$  and the already mapped  $\hat{X}_{Linj}^W$ , if the size of the projected  $X_{ObsLinj}^W$  in  $\hat{X}_{Linj}^W$ , is at least  $0.25/0.7 \approx 0.35$  times of the size of  $\hat{X}_{Linj}^W$ . Therefore, the overlapping parameters is equal to  $O_r = 1.35$ .

The parameter  $\delta_{dPnt}$  is used in the association of feature points. As, it is considered that at least an overlapping of 0.25 metres is needed for the association of two linear segments, it is also considered that a distance of 0.25 metres should be the maximum distance between points, allowing their association.

Finally, imagine a person walking in the navigation area, during the *pre-localization*, with at least 0.3 m/s of speed. This person will be "extracted" as an point or a linear segment feature. As this person is a dynamic "object", it is necessary to avoid its insertion in the state

vector. Therefore, MinScore should be such value that, after MinScore iterations, its distance in relation to the first appearance is higher than the  $\max(\delta_{\perp_{Lin}}, \delta_{d_{Pnt}}) = 0.5$  metres . Therefore MinScore can be computed using the following expression.

$$0.3 \text{ CycleTime} \cdot \text{MinScore} > 0.5 \quad (7.110)$$

where CycleTime is equal to 100 milliseconds. Therefore, MinScore need to be higher than:

$$\text{MinScore} > \frac{0.5}{0.3 \times 0.1} \approx 17 \text{ iteration} \quad (7.111)$$

The chosen value for MinScore was 20 iterations.

Update	
$K\sigma_\rho$ (mtres)	0.2
$K\sigma_\lambda$ (radians/degrees)	0.26/15
$K\sigma_x, K\sigma_y$ (metres)	0.15

Table 7.1 Parameters of the update in *Pre-localization*.

Association			
$\delta_{d_{Pnt}}$ (metres)	0.25	$\delta_{\perp_{Lin}}$ (metres)	0.5
$\delta_{\lambda_{Lin}}$ (radians/degrees)	0.78/45	$O_r$	1.35
<i>MinScore</i>		20	

Table 7.2 Parameters of association in *Pre-localization*.



## 8. Mapping

After achieved the vehicle's 2D location, using the *pre-localization* process, it is possible to create a 3D map of the surrounding environment (*mapping*).

Therefore, the three-dimensional world is represented as follows: a map  $M$ , which is divided into cells  $c_i$  (cube shaped) forming an occupancy grid. The occupancy grid has maximum width ( $x_{max}$ ), length ( $y_{max}$ ) and height ( $z_{max}$ ). The resolution (*res*) defines the size of each cell.

The matrix  $M$  at each position in world referential ( $x^W, y^W, z^W$ ) has two different values: the probability of being occupied,  $M(x^W, y^W, z^W) . p$ , and its state, ('occupied', 'free' or 'unknown'),  $M(x^W, y^W, z^W) . s$ .

The probability of the occupancy grid, needs to be filled using rules with the ability of highlight cells with high degree of certainty to be occupied. On the contrary, the rule must have the capacity of despise cells, whose probability to be occupied is lower.

The Bayes Rules have the capacity to perform the *mapping*, thus meeting the characteristics described in the previous paragraph. The Bayes rules applied in the 3D mapping is described in the following sub-section (8.1).

### 8.1. Bayes Rules

The probability that a cell  $c_i$  has of being occupied is defined as  $P(E)$ . Therefore,  $P(E/O)$  represents the probability that the cell has of being occupied if there is an observation corresponding to that cell  $c_i$ . On the contrary, the probability that a cell has of being occupied if there is no observation, is represented by  $P(E/\bar{O})$ .

The probability of the cell  $c_i$ , being observed if it is occupied, is represented by  $P(O/E)$ . On the other hand, the probability of an observation when the cell is not occupied is equal to  $P(O/\bar{E})$ . These two last probabilities are characteristic of the LRF.

In [50], experimental results are conducted using the *Hokuyo URG-04LX-UG01*, the same that was used in this work. During these experiments the values of  $P(O/E) = 0.9$  and  $P(O/\bar{E}) = 0.05$  were found. The same probabilities of  $P(O/E)$  and  $P(O/\bar{E})$  were used in the work described in this document.

At each sample, with the knowledge of the vehicle position, it is possible to obtain, in the world frame, a set of occupied cells  $o_i$ . Each cell  $o_i$  is corresponding to one laser scan point, with coordinates ( $x^W, y^W, z^W$ ), in the world referential.

The coordinates of the cell  $o_i = (x^W, y^W, z^W)$  has a corresponding cell  $c_i$  in the occupancy grid, defined as  $M$ .

For each measured occupied cell  $o_i$  is possible to obtain a set of cells called  $beam_i$ , which contains the occupied cell  $o_i$ , represented in blue in Fig. 8.1 and the free cells, the ones inside

the range of reading  $i$ ,  $\{f_{1i} \dots f_{ni}\}$  represented at grey in Fig. 8.1. Therefore, the set  $beam_i$  can be written as follows:

$$beam_i = \{o_i, f_{1i} \dots f_{ni}\} \quad (8.1)$$

Thus, to each set of cells belonging to the  $beam_i$ , it is necessary to compute:

- 1) For the cell  $c_i$  corresponding to  $o_i$ , the probability to be occupied if there is an observation ( $P(E/O)$ ).
- 2) For all cells  $c_i$  corresponding to the ones considered free  $\{f_{1i} \dots f_{ni}\}$ , it is necessary to compute the probability of being occupied if there is no observation ( $P(E/\bar{O})$ ).

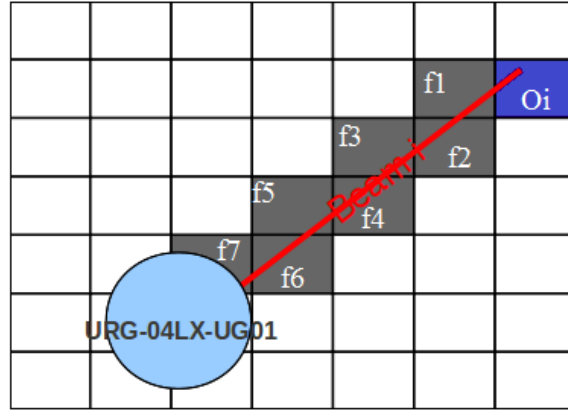


Fig. 8.1. Beam  $i$ . Occupied cell represented in blue ( $o_i$ ). Free cells represented in grey,  $\{f_{1i} \dots f_{ni}\}$ .

Both these probabilities,  $P(E/O)$  and  $P(E/\bar{O})$ , are computed using the following expressions:

$$P(E/O) = \frac{P(O/E) \cdot P(E)}{P(O/E) \cdot P(E) + P(O/\bar{E}) \cdot P(\bar{E})} \quad (8.2)$$

$$P(E/\bar{O}) = \frac{P(\bar{O}/E) \cdot P(E)}{P(\bar{O}/E) \cdot P(E) + P(\bar{O}/\bar{E}) \cdot P(\bar{E})} \quad (8.3)$$

The probabilities (8.2) and (8.3) are presented according to the Bayes rules, in which  $P(\bar{E}) = 1 - P(E)$ ,  $P(\bar{O}/E) = 1 - P(O/E)$  and  $P(\bar{O}/\bar{E}) = 1 - P(O/\bar{E})$ .

The probability  $P(E)$  represents the previous probability of the cell  $c_i$ . The initial value of  $P(E)$  in each cell is 0.5, because those cells are also likely to be occupied or free. The initial state of each cell  $c_i$  is 'unknown'. Therefore, initially, all the cell  $c_i$ , will have a probability equal to  $M(x^W, y^W, z^W).p = 0.5$ , while the state will be equal to the label  $M(x^W, y^W, z^W).s = \text{'unknown'}$ .

The pseudo-code of Algorithm 8.1, describes the application of the Bayes Rules over the set of cells belonging to the  $beam_i$ . The cells with a probability higher or equal than  $Pmin$  are classified as 'occupied' cells. On the contrary, the cells with probability lower than  $Pmin$  are considered 'free' cells.

S. Jia *et al.* in the work [50], uses the probability of 0.7 as value for  $Pmin$ . The same value was used, with good results, in the *mapping* process described in this Chapter, aiming

the 3D mapping of the surrounding environment. Thus, only the cells with probability higher than are considered 0.7 occupied, while the others are free.

Algorithm 8.1 Application of the Bayes Rules over the set of cells belonging to the  $beam_i$ .

```

for each cell of  $beam_i$  do

     $P(E) \leftarrow M(x^W, y^W, z^W).p$ 

     $P(\bar{E}) \leftarrow 1 - P(E)$ 

    if cell in  $\{f_{1i} \dots f_{ni}\}$  then

        apply equation 8.3

         $M(x^W, y^W, z^W).p \leftarrow P(E/\bar{O})$ 

    else // (cell =  $c_i$ )

        apply equation 8.2

         $M(x^W, y^W, z^W).p \leftarrow P(E/O)$ 

    end if

    if  $M(x^W, y^W, z^W).p \geq Pmin$  then

         $M(x^W, y^W, z^W).s \leftarrow 'occupied'$ 

    else

         $M(x^W, y^W, z^W).s \leftarrow 'free'$ 

    end if

end for

```

## 8.2. Creating Distance and Gradient Matrices

After obtained the occupancy grid  $M$ , in the three-dimensional space, through the application of the Bayes Rules, it is necessary to compute the distance and gradient matrices. The distance and gradient matrices are the core base of the localization 3D Matching algorithm described in the next Chapter, Chapter 9.

In fact, these matrices, need to be pre-computed and stored in memory to be used during the *localization* process, in the vehicle normal operation. The pre-computation of the distance and gradient matrices guarantee a fast performance of the 3D Matching algorithm.

The need and utility of these matrices will be described with more care in Chapter 9. At the moment its only important to refer that in this sub-section will be explained how to compute the distance and the gradient matrices through the already obtained 3D occupancy grid  $M$ .

These matrices have equal dimension and resolution of the occupancy grid  $M$ : width ( $x_{max}$ ), length ( $y_{max}$ ), height ( $z_{max}$ ) and resolution ( $res$ ).

The distance matrix, represented by  $dmap$ , has at each coordinate position in the world frame  $(x^W, y^W, z^W)$ , the distance of the nearest 'occupied' cell, in the occupancy grid  $M$ .

The gradient matrix  $\nabla x_{3D}$  is computed in order to the  $x$  direction. This matrix represent the variation of  $dmap$  in each coordinate position  $(x^W, y^W, z^W)$ , with the variation of  $x$ . In a similar way, the gradient matrix  $\nabla y_{3D}$ , contains the variation of  $dmap$  in the position  $(x^W, y^W, z^W)$ , but with the variation of  $y$ .

The algorithm used to compute the distance matrix is described as follows:

1) It is necessary to initialize the distance matrix. Therefore, it is visited all the cells  $c_i$  of  $M$ . The cells in  $M$  classified as 'occupied' cells, have a corresponding distance matrix value, equal to zero. The others, classified as 'unknown' or 'free', have a corresponding value in the distance matrix equal to infinite, which is represented by a large number  $(x_{max} \cdot y_{max} \cdot z_{max})$ .

2) After initialized, the computation of the distance matrix is constituted by two sub-steps:

2.1) The first is performed visiting the distance matrix from the upper and left corner, updating the actual distance map position based on the previous neighbours.

2.2) The second step is completed by updating the distance matrix, using the next neighbours, when each position of the distance map is visited, starting from the bottom and right corner.

Considering the 3D space, the step 1) described above of the algorithm to compute the distance matrix ( $dmap$ ) is presented in the pseudo-code shown in Algorithm 8.2. The step 2) of this algorithm, as described above, is presented in the Algorithm 8.3. For more details about the Distance Transform see the works [54] and [55].

Algorithm 8.2 Distance transform applied to the occupancy grid  $M$ . Initialization of each cell. Step number 1).

```

for  $x^W \leftarrow 1$  to  $x_{max}$  do
  for  $y^W \leftarrow 1$  to  $y_{max}$  do
    for  $z^W \leftarrow 1$  to  $z_{max}$  do
      if  $M(x^W, y^W, z^W).s = \text{'occupied'}$  then
         $dmap(x^W, y^W, z^W) \leftarrow 0$ 
      else
         $dmap(x^W, y^W, z^W) \leftarrow x_{max} \cdot y_{max} \cdot z_{max}$ 
      end if
    end for
  end for
end for

```

Algorithm 8.3 Distance transform applied to the occupancy grid  $M$ . Step number 2).

```

for  $x^W \leftarrow 1$  to  $xmax$  do

    for  $y^W \leftarrow 1$  to  $y_{max}$  do

        for  $z^W \leftarrow 1$  to  $z_{max}$  do

             $h \leftarrow \min(dmap(x^W, y^W, z^W), dmap(x^W - 1, y^W, z^W) + res)$ 

             $h \leftarrow \min(h, dmap(x^W, y^W - 1, z^W) + res)$ 

             $h \leftarrow \min(h, dmap(x^W, y^W, z^W - 1) + res)$ 

             $h \leftarrow \min(h, dmap(x^W - 1, y^W - 1, z^W) + \sqrt{2} \cdot res)$ 

             $h \leftarrow \min(h, dmap(x^W - 1, y^W, z^W - 1) + \sqrt{2} \cdot res)$ 

             $h \leftarrow \min(h, dmap(x^W, y^W - 1, z^W - 1) + \sqrt{2} \cdot res)$ 

             $h \leftarrow \min(h, dmap(x^W - 1, y^W - 1, z^W - 1) + \sqrt{3} \cdot res)$ 

             $dmap(x^W, y^W, z^W) \leftarrow h$ 

        end for

    end for

end for

for  $x^W \leftarrow xmax$  to  $1$  do

    for  $y^W \leftarrow y_{max}$  to  $1$  do

        for  $z^W \leftarrow z_{max}$  to  $1$  do

             $h \leftarrow \min(dmap(x^W, y^W, z^W), dmap(x^W + 1, y^W, z^W) + res)$ 

             $h \leftarrow \min(h, dmap(x^W, y^W + 1, z^W) + res)$ 

             $h \leftarrow \min(h, dmap(x^W, y^W, z^W + 1) + res)$ 

             $h \leftarrow \min(h, dmap(x^W + 1, y^W + 1, z^W) + \sqrt{2} \cdot res)$ 

             $h \leftarrow \min(h, dmap(x^W + 1, y^W, z^W + 1) + \sqrt{2} \cdot res)$ 

             $h \leftarrow \min(h, dmap(x^W, y^W + 1, z^W + 1) + \sqrt{2} \cdot res)$ 

             $h \leftarrow \min(h, dmap(x^W + 1, y^W + 1, z^W + 1) + \sqrt{3} \cdot res)$ 

             $dmap(x^W, y^W, z^W) \leftarrow h$ 

        end for

    end for

end for

```

Once the distance matrix ( $dmap$ ) was calculated, the gradients ( $\nabla x_{3D}$ ,  $\nabla y_{3D}$ ) are computed using the Sobel Filter. For more details about the Sobel Filter see the reference [56]. The gradient  $\nabla x_{3D}$  is equal to the weighted average of  $dmap$  using a vertical Sobel operator of the equation (8.4). The gradient  $\nabla y_{3D}$  is equal to the weighted average of  $dmap$  using the horizontal Sobel operator of the equation (8.5).

$$H = \begin{bmatrix} -1 & +0 & +1 \\ -2 & +0 & +2 \\ -1 & +0 & +1 \end{bmatrix} \quad (8.4)$$

$$V = \begin{bmatrix} -1 & -2 & -1 \\ +0 & +0 & +0 \\ +1 & +2 & +1 \end{bmatrix} \quad (8.5)$$

Therefore, for each point  $p$  with coordinates in the world referential ( $x^W, y^W, z^W$ ) and with distance value equal to  $dmap(p) = d$ , the neighbours are represented by the following matrix:

$$N(x, y, z) = \begin{bmatrix} lo & mo & ro \\ lm & d & rm \\ lu & mu & ru \end{bmatrix} \quad (8.6)$$

The gradients for  $x$  and  $y$ , in the two-dimensional space, are calculated using the following expressions:

$$\nabla x(x^W, y^W, z^W) = \frac{H \times N(x^W, y^W, z^W)}{8} \quad (8.7)$$

$$\nabla y(x^W, y^W, z^W) = \frac{V \times N(x^W, y^W, z^W)}{8} \quad (8.8)$$

To compute the  $x$  gradient  $\nabla x_{3D}$ , in the three-dimensional space, to each cell of the gradient matrix, with coordinates ( $x^W, y^W, z^W$ ), it is applied the equation (8.9). The gradient in the  $y$  direction,  $\nabla y_{3D}$ , is computed with the equation (8.10), applying it to each cell of the distance matrix.

$$\nabla x_{3D}(x^W, y^W, z^W) = \frac{\nabla x(x^W, y^W, z^W - 1) + \nabla x(x^W, y^W, z^W) + \nabla x(x^W, y^W, z^W + 1)}{3} \quad (8.9)$$

$$\nabla y_{3D}(x^W, y^W, z^W) = \frac{\nabla y(x^W, y^W, z^W - 1) + \nabla y(x^W, y^W, z^W) + \nabla y(x^W, y^W, z^W + 1)}{3} \quad (8.10)$$

## 9. Localization

The *three-dimensional map-based* localization methodology implemented and presented in this document is divided into two crucial parts: 1) *pre-localization* and *mapping*, already described in Chapters 7 and 8; 2) *localization*.

The *localization* part corresponds to the normal operation of the vehicle and it is performed using a 3D matching algorithm. The localization estimates the vehicle 2D position and orientation ( $x_v$ ,  $y_v$  and  $\theta_v$ ), performing a matching using data in the three-dimensional space, i.e. laser points with coordinates ( $x$ ,  $y$  and  $z$ ).

Using a known 3D map, it is possible to build (offline) a matrix of distances. At each coordinate point, this matrix includes the distance to the closest occupied cell. It is also possible to compute the gradient matrices in the  $x$  and  $y$  directions, which represent the variation of the distance according to these directions. The computation of these matrices is described in the sub-section 8.2.

The gradient matrices help to obtain the convergence direction for the optimisation algorithm. These matrices are the core of the matching algorithm presented here since they are look-up tables, which reduces the need to perform calculations and decreases the execution time needed. These matrices are stored in memory to be used during the 3D matching algorithm. The optimisation algorithm used was the Resilient Back-Propagation (RPROP), which is a gradient descent method [69].

For an easier representation, the Fig. 9.1, Fig. 9.2 and Fig. 9.3 shows the two-dimensional distance and gradient matrices about a corridor.

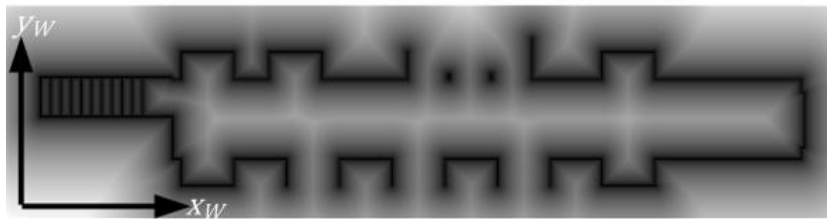


Fig. 9.1 Maps on a corridor where experiments were conducted. Distance Matrix.

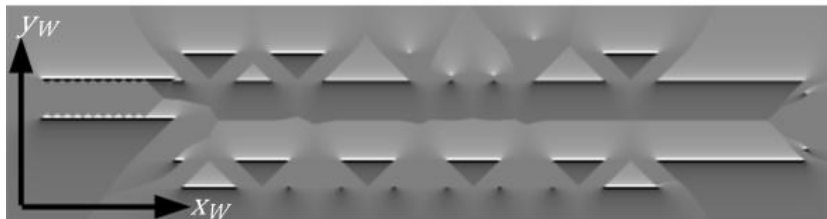


Fig. 9.2 Maps on a corridor where experiments were conducted. Gradient Matrix. Distance variation in direction  $y$ .

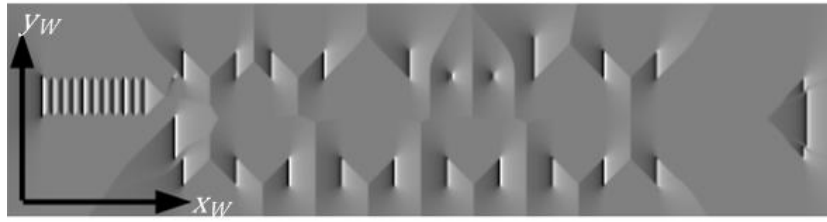


Fig. 9.3 Maps on a corridor where experiments were conducted. Gradient Matrix. Distance variation in direction x.

The matching algorithm described in this document performed using 3D data, takes the vehicle from its previous estimated pose and finds the best matching in relation to the stored map. For that, the algorithm uses the actual set of data acquired by the *observation module*, minimizing the cost function defined in equation (9.13).

As already referred the algorithm of localization presented here is based on the Perfect Match algorithm described by M. Lauer *et al.* [4].

This algorithm was implemented in the 5DPO team's robots to be used during the robotic soccer Middle Size League games. In this robots the algorithm uses a matching with 2D data acquired with omnidirectional vision. But, the Perfect Match algorithm was transformed to be used as *localization* process in the RobVigil. The three crucial alterations and improvements made were:

1) Instead using artificial vision, which implies an heavy image pre-processing, it is used Laser Range Finder information, acquired with a tilting LRF called *observation module*. The use of this *observation module*, allows obtaining a huge amount of three-dimensional data, about the surrounding environment, without a heavy cost when pre-processing the raw data, to extract helpful information, as it occurs with the artificial vision.

2) The vehicle's pose estimation is still done in the two-dimensional space, while the data used is about the three-dimensional, maintaining the low computational needs. The use of three-dimensional data in this work has an advantage, since it allows the use of the headroom map (upper side of a building - above 1.8 metres ), which remains equal during large periods of time, and can be considered static, to perform the algorithm of localization.

The objective is the robot operation inside indoor buildings as service facilities (for instance shopping malls), with lot of people "travelling". When the upper side of a building is used, the 3D matching will reach a better performance, since the acquired data is not affected with the persons that are crossing the navigation area.

3) Finally, M. Lauer *et al.* [4] uses a stochastic weighted averaging method (which is a simplified Kalman filter) to merge the vehicle's pose estimation given by odometry and the Perfect Match algorithm. But the stochastic weighted averaging ignores the crossed variances (covariances) and therefore, despite the error dependence between the vehicle state variables ( $x_v, y_v$  and  $\theta_v$ ).

Therefore, the *localization* algorithm proposed in this document, uses an Extended Kalman Filter (EKF), aiming to perform the fusion between the pose estimation given by odometry and the 3D Perfect Match. The EKF enters in account with the full covariance matrix and models in a better way the vehicle's error accumulated during its displacement.

Furthermore, the EKF fusion is important to increase the precision and robustness of the localization algorithm described. A wrong 3D Perfect Match behaviour will not make the



localization to lose the vehicle's position tracking, since the fusion with odometry will avoid the jump of the vehicle's pose estimation. Also, the EKF operation, becomes the vehicle's localization estimation smother allowing a better trajectory control.

The entire three-dimensional Matching algorithm is described with more detail in the next sub-section (9.1).

The execution time needed in this localization algorithm is proportional to the number of points (data) used to perform the 3D Perfect Match. Therefore, the time spent in the localization algorithm, can be characterized and maximized, since it is possible to limit the number of the used points.

### 9.1. Three-Dimensional Matching

The 3D matching localization algorithm is described as the pseudo-code shown in Algorithm 9.1. The main cycle has three crucial stages: i) Kalman Filter Prediction; ii) 3D Perfect Match procedure. iii) Kalman Filter Update. The Kalman Filter equations can be seen in more detail in [53]. The three stages are detailed explained in the following three sub-sections.

Algorithm 9.1 3D Matching Localization Cycle.

<i>Cycle(Odo, PntList<sup>L</sup>)</i>
$\hat{X}_v(k+1 k), P_v(k+1 k) \leftarrow \text{KalmanPrediction}(\hat{X}_v(k k), P_v(k k), Odo)$
$X_{Match}, P_{Match} \leftarrow \text{3DPerfectMatch}(\hat{X}_v(k+1 k), P_v(k+1 k), PntList^R)$
$\hat{X}_v(k+1 k+1), P_v(k+1 k+1) \leftarrow \text{KalmanUpdate}(\hat{X}_v(k+1 k), P_v(k+1 k), X_{Match}, P_{Match})$

#### 9.1.1. Kalman Filter Prediction

The Kalman Filter Prediction stage takes the previously estimated vehicle state  $\hat{X}_v(k|k)$  and, using odometry, estimates the next vehicle state  $\hat{X}_v(k+1|k)$ . The real vehicle state is represented by the following vector, set up by variables  $x_v, y_v$  and  $\theta_v$  (2D coordinates and orientation in relation to the x direction), as shown in Fig. 7.3:

$$X_v = [x_v \quad y_v \quad \theta_v]^T \quad (9.1)$$

The vehicle estimated state, along this chapter, will be represented as shown in the following vector:

$$\hat{X}_v = [\hat{x}_v \quad \hat{y}_v \quad \hat{\theta}_v]^T \quad (9.2)$$

Aiming to model the vehicle's kinematic movement, the following transition function  $f_v(X_v(k|k), Odo, q_v)$  was used, [62]:

$$f_v(X_v(k|k), Odo, q_v) = \begin{bmatrix} x_v(k|k) + d \cdot \cos\left(\theta_v(k|k) + \frac{\Delta\theta}{2}\right) + \varepsilon_{qx} \\ y_v(k|k) + d \cdot \sin\left(\theta_v(k|k) + \frac{\Delta\theta}{2}\right) + \varepsilon_{qy} \\ \theta_v(k|k) + \Delta\theta + \varepsilon_{q\theta} \end{bmatrix} \quad (9.3)$$

in which the odometry input,  $Odo$ , is given for the following vector, composed by the variables  $d$  and  $\Delta\theta$ , which represent the estimated linear and angular displacements, respectively, calculated as described in equations (5.26) and (5.27), through the information acquired by the odometers:

$$Odo = \begin{bmatrix} d \\ \Delta\theta \end{bmatrix} \quad (9.4)$$

The kinematic model's error  $q_v$ , was modelled as additive Gaussian noise with zero mean and covariance  $Q_v$ . The vector  $q_v$ , represented as the following, is constituted by three errors, each one correspondent to the variables that are intended to estimate.

$$q_v = [\varepsilon_{qx} \quad \varepsilon_{qy} \quad \varepsilon_{q\theta}]^T \quad (9.5)$$

As the noise error  $q_v$  is modelled as Gaussian noise with zero mean, then its expected value is zero  $\hat{q}_v = 0$ . The new vehicle estimated state, after the prediction stage, appears equal to the following equation:

$$\begin{aligned} \hat{X}_v(k+1|k) &= f_v(\hat{X}_v(k|k), Odo) \Leftrightarrow \\ \hat{X}_v(k+1|k) &= \begin{bmatrix} \hat{x}_v(k|k) + d \cdot \cos\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \\ \hat{y}_v(k|k) + d \cdot \sin\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \\ \hat{\theta}_v(k|k) + \Delta\theta \end{bmatrix} \end{aligned} \quad (9.6)$$

The Kalman Filter Prediction stage, also uses the previous estimated covariance matrix  $P_v(k|k)$  and computes the next,  $P_v(k+1|k)$ . Therefore, the estimated covariance after the prediction stage,  $P_v(k+1|k)$ , is given by the equation:

$$P_v(k+1|k) = \frac{\partial f_v}{\partial X_v} P_v(k|k) \frac{\partial f_v^T}{\partial X_v} + \frac{\partial f_v}{\partial q_v} Q_v \frac{\partial f_v^T}{\partial q_v} \quad (9.7)$$

To compute the prediction covariance, it is necessary to calculate the gradient of the model transition function  $f_v(\hat{X}_v(k|k), Odo)$ :

$$\frac{\partial f_v}{\partial X_v} = \begin{bmatrix} 1 & 0 & -d \cdot \sin\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \\ 0 & 1 & +d \cdot \cos\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \\ 0 & 0 & 1 \end{bmatrix} \quad (9.8)$$

The gradient of the transition function  $f_v$ , in order to the noise  $q_v$ , equal to  $\frac{\partial f_v}{\partial q_v}$ , is the identity matrix. Then, the new estimative of the covariance matrix can be re-written as follows:

$$P_v(k+1|k) = \begin{bmatrix} 1 & 0 & -d \cdot \sin\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \\ 0 & 1 & +d \cdot \cos\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \\ 0 & 0 & 1 \end{bmatrix} P_v(k|k) \begin{bmatrix} 1 & 0 & -d \cdot \sin\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \\ 0 & 1 & +d \cdot \cos\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \\ 0 & 0 & 1 \end{bmatrix}^T + Q_v \quad (9.9)$$

The covariance  $Q_v$  depends on the estimative of the vehicle's distance and rotation ( $d$  and  $\Delta\theta$ ):

$$Q_v = \begin{bmatrix} \left[d \cdot \cos\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \cdot \sigma_d\right]^2 & 0 & 0 \\ 0 & \left[d \cdot \sin\left(\hat{\theta}_v(k|k) + \frac{\Delta\theta}{2}\right) \cdot \sigma_d\right]^2 & 0 \\ 0 & 0 & (d \cdot \sigma_{d\theta})^2 + (\Delta\theta \cdot \sigma_\theta)^2 \end{bmatrix} \quad (9.10)$$

When the vehicle moved 1 metre in front, the odometry, accumulated a translational and rotational error equal to  $\sigma_d$  and  $\sigma_{d\theta}$ , respectively. When the vehicle rotates 1 radian, the odometry error on the estimation of the vehicle rotation is equal to  $\sigma_\theta$ . These parameters were obtained as already described in Chapter 5.

Therefore, when the vehicle had no movement, the kinematic model error, represented by  $Q_v$ , is zero, as can be conclude with the equation (9.10).

On the contrary, the variance of the states  $x_v$  and  $y_v$ , equal to  $Q_v(1,1)$  and  $Q_v(2,2)$ , increase with the quadratic displacement of the vehicle in the x and y axis, respectively. The variance of  $\theta_v$ , equal to  $Q_v(3,3)$ , increase with the quadratic translational and rotational displacement.

Therefore, the Kalman Filter Prediction stage, can be summarized with the following equations.

Algorithm 9.2 Equations of the Kalman Filter Prediction stage.

<i>KalmanPrediction</i> ( $\hat{X}_v(k k), P_v(k k), Odo$ )
$\hat{X}_v(k+1 k) \leftarrow f_v(\hat{X}_v(k k), Odo)$
$P_v(k+1 k) \leftarrow \frac{\partial f_v}{\partial X_v} P_v(k k) \frac{\partial f_v}{\partial X_v}^T + Q_v$

### 9.1.2. 3D Perfect Match

The 3D Perfect Match procedure takes the vehicle's state propagation provided by the Kalman Filter Prediction stage and performs the optimization routine described in the pseudo-code depicted in Algorithm 9.3. This procedure has three crucial steps:

- i) Computing the gradient matrix: *GradientMatrix*;
- ii) Optimization routine, Resilient Back-Propagation [69]: *RPROP*;
- iii) Compute the resultant second derivative, i.e. computation of the covariance matrix of the 3D Perfect Match: *SecondDerivative*.

The initial state of the vehicle to be used in the 3D Perfect Match algorithm is given by the vehicle pose propagated by the Kalman Filter Prediction stage:  $X_{Match} = \hat{X}_v(k+1|k)$ .

The state vector  $X_{Match}$  is step up by the variables  $x_{Match}$ ,  $y_{Match}$  and  $\theta_{Match}$ , which has a direct correspondence with the vehicle state variables  $x_v$ ,  $y_v$  and  $\theta_v$ .

The first two steps, *MatchError* and *RPROP*, are continuously iterated until the maximum number of iterations (*iterMax*) is reached. The last and third step is performed only once, after obtained the result of the vehicle state.

At each iteration, the list of points obtained by the tilting Laser Range Finder, represented in the vehicle frame  $PList^V$ , passed as parameter, need to be computed in the world referential using the previous state of the 3D Perfect Match algorithm.

Algorithm 9.3 3D Perfect Match Cycle.

<b><i>PerfectMatch</i></b> ( $X_v(k+1 k), P_v(k+1 k), PList^V$ )
$X_{Match} \leftarrow X_v(k+1 k)$ $\nabla E(0) \leftarrow 0$ <b>for</b> $n \leftarrow 1$ <b>to</b> $iterMax$ <b>do</b> $PList^W \leftarrow X_{Match} + R_V^W \cdot PList^V$ $\nabla E(n) \leftarrow GradientMatrix(PList^W, X_{Match})$ $X_{Match} \leftarrow RPROP(\nabla E(n), \nabla E(n-1), X_{Match})$ <b>end for</b> $P_{Match} \leftarrow SecondDerivarive(\nabla E(n), PList^W)$

Each point  $PList(i)^V$  has coordinates in the robot referential equal to  $(x_i^V, y_i^V, z_i^V)$ . The correspondence of this point in the world referential, was already explained in Chapter 5. Nevertheless, lets remember this relation.  $PList(i)^W$  has coordinates  $(x_i^W, y_i^W, z_i^W)$ , which are given by the following expression:

$$\begin{bmatrix} x_i^W \\ y_i^W \\ z_i^W \end{bmatrix} = \begin{bmatrix} x_{Match} \\ y_{Match} \\ \theta_{Match} \end{bmatrix} + R_V^W \cdot \begin{bmatrix} x_i^V \\ y_i^V \\ z_i^V \end{bmatrix} \quad (9.11)$$

in which the  $R_V^W$  matrix represents the rotation of the vehicle in relation to the world referential, expressed in the world referential:

$$R_R^W = \begin{bmatrix} \cos \theta_{Match} & -\sin \theta_{Match} & 0 \\ \sin \theta_{Match} & \cos \theta_{Match} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (9.12)$$

The  $dmap$ ,  $\nabla x$  and the  $\nabla y$  matrices were used to compute the gradient matrix about the cost function, corresponding to the list of points  $PList^W$ . This gradient is used by the *RPROP* step, to obtain the direction of minimization of the cost function.

The cost function of the list of points of the Laser Range Finder scan ( $PList^W$ ), is determined with the following equations:

$$E = \sum_{i=1}^N E_i, \quad E_i = 1 - \frac{L_c^2}{L_c^2 + d_i^2} \quad (9.13)$$

in which  $d_i$  is representative of the *dmap* and cost function values, which correspond to the point  $PList(i)^W$ , with coordinates  $(x_i^W, y_i^W, z_i^W)^T$ .  $N$  is the number of points in the list of three-dimensional points  $PList^W$ .

The cost function  $E_i$  was designed to achieve a behaviour similar to the quadratic error function for the case of small values of  $d_i$ , neglecting points with large values of  $d_i$  (outliers). In order for the cost function to meet the condition  $0.5 \leq E_i \leq 1$ , when the distance of the point  $PList(i)^W$  is equal or higher than 1 metres,  $d_i \geq 1 \text{ metre}$ , the value of  $L_c$  used in the work described in this document was 1 metre.

The gradient of the cost function, in order to the state  $X_{Match}$  is given by the following expression:

$$\begin{aligned} \nabla E(n) &= [\nabla E_x \quad \nabla E_y \quad \nabla E_\theta]^T \Leftrightarrow \\ \nabla E(n) &= \left[ \frac{\partial E}{\partial x_{Match}} \quad \frac{\partial E}{\partial y_{Match}} \quad \frac{\partial E}{\partial \theta_{Match}} \right]^T \Leftrightarrow \\ \nabla E(n) &= \left[ \sum_{i=1}^N \frac{\partial E_i}{\partial x_{Match}} \quad \sum_{i=1}^N \frac{\partial E_i}{\partial y_{Match}} \quad \sum_{i=1}^N \frac{\partial E_i}{\partial \theta_{Match}} \right]^T \Leftrightarrow \\ \nabla E(n) &= \sum_{i=1}^N \frac{2L_c^2 \cdot d_i}{(L_c^2 + d_i^2)^2} \left[ \frac{\partial d_i}{\partial x_{Match}} \quad \frac{\partial d_i}{\partial y_{Match}} \quad \frac{\partial d_i}{\partial \theta_{Match}} \right]^T \Leftrightarrow \\ \nabla E(n) &= \sum_{i=1}^N \frac{2L_c^2 \cdot d_i}{(L_c^2 + d_i^2)^2} \frac{\partial d_i}{\partial X_{Match}} \end{aligned} \quad (9.14)$$

The partial derivative vector  $\frac{\partial d_i}{\partial X_{Match}}$  is obtained using the following expression:

$$\frac{\partial d_i}{\partial X_{Match}} = \left[ \frac{\partial d_i}{\partial x_i^W} \cdot \frac{\partial x_i^W}{\partial x_{Match}}, \frac{\partial d_i}{\partial y_i^W} \cdot \frac{\partial y_i^W}{\partial y_{Match}}, \frac{\partial d_i}{\partial x_i^W} \cdot \frac{\partial x_i^W}{\partial \theta_v} + \frac{\partial d_i}{\partial y_i^W} \cdot \frac{\partial y_i^W}{\partial \theta_{Match}} \right]^T \quad (9.15)$$

Using the equations presented at (9.11) and (9.12), the following expressions can be written:

$$\frac{\partial x_i^W}{\partial x_{Match}} = 1, \quad \frac{\partial d_i}{\partial x_i^W} = \nabla x(x_i^W, y_i^W, z_i^W) \quad (9.16)$$

$$\frac{\partial y_i^W}{\partial y_{Match}} = 1, \quad \frac{\partial d_i}{\partial y_i^W} = \nabla y(x_i^W, y_i^W, z_i^W) \quad (9.17)$$

$$\frac{\partial x_i^W}{\partial \theta_{Match}} = -\sin \theta_{Match} \cdot x_i^W - \cos \theta_{Match} \cdot y_i^W \quad (9.18)$$

$$\frac{\partial y_i^W}{\partial \theta_{Match}} = \cos \theta_{Match} \cdot x_i^W - \sin \theta_{Match} \cdot y_i^W \quad (9.19)$$

in which  $\nabla x(x_i^W, y_i^W, z_i^W)$  and  $\nabla y(x_i^W, y_i^W, z_i^W)$  are the gradient values of the pre-computed and stored gradient matrices, at the position  $(x_i^W, y_i^W, z_i^W)$ , in the world frame. Therefore, the vector components on equation (9.15) can be re-written as the following expressions:

$$\frac{\partial d_i}{\partial x_{Match}} = \nabla x(x_i^W, y_i^W, z_i^W) \quad (9.20)$$

$$\frac{\partial d_i}{\partial y_{Match}} = \nabla y(x_i^W, y_i^W, z_i^W) \quad (9.21)$$

$$\frac{\partial d_i}{\partial \theta_{Match}} = \begin{bmatrix} \nabla x(x_i^W, y_i^W, z_i^W) \\ \nabla y(x_i^W, y_i^W, z_i^W) \end{bmatrix}^T \begin{bmatrix} -\sin \theta_{Match} & -\cos \theta_{Match} \\ \cos \theta_{Match} & -\sin \theta_{Match} \end{bmatrix} \begin{bmatrix} x_i^W \\ y_i^W \end{bmatrix} \quad (9.22)$$

Enter in the consideration that, the matching error is a measurement about how far the actual estimative of vehicle pose ( $X_{Match}$ ) is, from the true one, i.e. it measures the distance that the list of points  $PList$  is from the obstacles on the 3D occupancy grid. In summary, the matching error computation can be performed applying the equation (9.13). The gradient matrix about the cost function in order to the vehicle state is given by a vector shown in the following expression:

$$\nabla E(n) = \sum_{i=1}^N \frac{2L_c^2 \cdot d_i}{(L_c^2 + d_i^2)^2} \begin{bmatrix} \nabla x(x_i^W, y_i^W, z_i^W) \\ \nabla y(x_i^W, y_i^W, z_i^W) \\ \begin{bmatrix} \nabla x(x_i^W, y_i^W, z_i^W) \\ \nabla y(x_i^W, y_i^W, z_i^W) \end{bmatrix}^T \begin{bmatrix} -\sin \theta_{Match} & -\cos \theta_{Match} \\ \cos \theta_{Match} & -\sin \theta_{Match} \end{bmatrix} \begin{bmatrix} x_i^W \\ y_i^W \end{bmatrix} \end{bmatrix} \quad (9.23)$$

After computing the gradient matrix, it is possible to apply the RPROP algorithm to each vehicle state variable. The algorithm takes the previous state of the vehicle, equal to  $X_{Match}$  and estimates a new position for the vehicle, which is used in the next iteration.

Below it is done the explanation of the optimization algorithm RPROP. This explanation will be done for the estimation of the state variable  $x_{Match}$ . But the RPROP algorithm, is applied for the estimation of the others state variables,  $y_{Match}$  and  $\theta_{Match}$ , in a similar way, as it is presented in the pseudo-code of the Algorithm 9.4.

If the actual gradient  $\nabla E_x(n)$  is different from zero then the following steps are performed:

1) If the product  $\nabla E_x(n) \times \nabla E_x(n-1)$  is lower than zero, it means that the algorithm already passes a local minima, and then the direction of the convergence need to be inverted. In that case the actual step  $\lambda_{x_{Match}}$  will be lower than the previous, aiming to decelerate the convergence to find the nearest local minimum:

$$\lambda_{x_{Match}} = \lambda_{x_{Match}} \times \eta_{x_{Match}}^-, \quad \eta_{x_{Match}}^- \in ]0,1[ \quad (9.24)$$

2) If the product  $\nabla E_x(n) \times \nabla E_x(n-1)$  is higher than zero, it means that the algorithm is still converging to the local minima, and then the direction of the convergence should be maintained. In that case, the actual step  $\lambda_{x_{Match}}$  will be higher than the previous, aiming to accelerate the convergence:

$$\lambda_{x_{Match}} = \lambda_{x_{Match}} \times \eta_{x_{Match}}^+, \quad \eta_{x_{Match}}^+ \geq 1 \quad (9.25)$$

3) If the actual gradient  $\nabla E_x(n)$ , is higher than zero, it means that the cost function  $E$ , is increasing in the positive direction of  $x_{Match}$ . Therefore, the previous value of  $x_{Match}$ , should be decreased with the actual step  $\lambda_{x_{Match}}$ , as is shown in the equation:

$$x_{Match} = x_{Match} - \lambda_{x_{Match}} \quad (9.26)$$

4) On the contrary, if the derivative is lower than zero, the cost function  $E$ , is decreasing in the positive direction of  $x_{Match}$ . In this case, the present value of  $x_{Match}$  should be obtained, adding to its previous value, the actual step  $\lambda_{x_{Match}}$ , as shown in the following equation:

$$x_{Match} = x_{Match} + \lambda_{x_{Match}} \quad (9.27)$$

Algorithm 9.4 Resilient Back-Propagation algorithm. For each variable that is intended to estimate,  $x_{Match}$ ,  $y_{Match}$  and  $\theta_{Match}$ , the same algorithm runs in a parallel way.

<b><i>RPROP</i>(<math>\nabla E(n), \nabla E(n-1), X_{Match}</math>)</b>		
<b><i>if</i>(<math>\nabla E_x(n) \neq 0</math>) <i>then</i></b> <b><i>if</i>(<math>\nabla E_x(n) \cdot \nabla E_x(n-1) &gt; 0</math>) <i>then</i></b> $\lambda_{x_{Match}} \leftarrow \lambda_{x_{Match}} \cdot \eta_{x_{Match}}^+$ <b><i>end if</i></b> <b><i>if</i>(<math>\nabla E_x(n) \cdot \nabla E_x(n-1) &lt; 0</math>) <i>then</i></b> $\lambda_{x_{Match}} \leftarrow \lambda_{x_{Match}} \cdot \eta_{x_{Match}}^-$ <b><i>end if</i></b> <b><i>if</i>(<math>\nabla E_x(n) &gt; 0</math>) <i>then</i></b> $x_{Match} \leftarrow x_{Match} - \lambda_{x_{Match}}$ <b><i>end if</i></b> <b><i>if</i>(<math>\nabla E_x(n) &lt; 0</math>) <i>then</i></b> $x_{Match} \leftarrow x_{Match} + \lambda_{x_{Match}}$ <b><i>end if</i></b> <b><i>end if</i></b>	<b><i>if</i>(<math>\nabla E_y(n) \neq 0</math>) <i>then</i></b> <b><i>if</i>(<math>\nabla E_y(n) \cdot \nabla E_y(n-1) &gt; 0</math>) <i>then</i></b> $\lambda_{y_{Match}} \leftarrow \lambda_{y_{Match}} \cdot \eta_{y_{Match}}^+$ <b><i>end if</i></b> <b><i>if</i>(<math>\nabla E_y(n) \cdot \nabla E_y(n-1) &lt; 0</math>) <i>then</i></b> $\lambda_{y_{Match}} \leftarrow \lambda_{y_{Match}} \cdot \eta_{y_{Match}}^-$ <b><i>end if</i></b> <b><i>if</i>(<math>\nabla E_y(n) &gt; 0</math>) <i>then</i></b> $y_{Match} \leftarrow y_{Match} - \lambda_{y_{Match}}$ <b><i>end if</i></b> <b><i>if</i>(<math>\nabla E_y(n) &lt; 0</math>) <i>then</i></b> $y_{Match} \leftarrow y_{Match} + \lambda_{y_{Match}}$ <b><i>end if</i></b> <b><i>end if</i></b>	<b><i>if</i>(<math>\nabla E_\theta(n) \neq 0</math>) <i>then</i></b> <b><i>if</i>(<math>\nabla E_\theta(n) \cdot \nabla E_\theta(n-1) &gt; 0</math>) <i>then</i></b> $\lambda_{\theta_{Match}} \leftarrow \lambda_{\theta_{Match}} \cdot \eta_{\theta_{Match}}^+$ <b><i>end if</i></b> <b><i>if</i>(<math>\nabla E_\theta(n) \cdot \nabla E_\theta(n-1) &lt; 0</math>) <i>then</i></b> $\lambda_{\theta_{Match}} \leftarrow \lambda_{\theta_{Match}} \cdot \eta_{\theta_{Match}}^-$ <b><i>end if</i></b> <b><i>if</i>(<math>\nabla E_\theta(n) &gt; 0</math>) <i>then</i></b> $\theta_{Match} \leftarrow \theta_{Match} - \lambda_{\theta_{Match}}$ <b><i>end if</i></b> <b><i>if</i>(<math>\nabla E_\theta(n) &lt; 0</math>) <i>then</i></b> $\theta_{Match} \leftarrow \theta_{Match} + \lambda_{\theta_{Match}}$ <b><i>end if</i></b> <b><i>end if</i></b>

The limitation of the number of iterations in the 3D Perfect Match routine makes it possible to guarantee a maximum time of execution for the algorithm. Therefore, as it is intended to operate online, it is necessary to ensure that this maximum time is lower than the observation module cycle time (100 milliseconds). The following experiment was conducted aiming to guarantee the convergence of the RPROP in the 3D Perfect Match algorithm to a solution, with a minimum number of iterations.

The values  $\eta_{x_{Match}}^+$ ,  $\eta_{y_{Match}}^+$ ,  $\eta_{\theta_{Match}}^+$ ,  $\eta_{x_{Match}}^-$ ,  $\eta_{y_{Match}}^-$ ,  $\eta_{\theta_{Match}}^-$  and the initial value of the steps  $\lambda_{x_{Match}}$ ,  $\lambda_{y_{Match}}$ ,  $\lambda_{\theta_{Match}}$  are empirical and adjustable parameters. These parameters of the RPROP algorithm were tested in the intervals  $\eta_{x_{Match}}^+$ ,  $\eta_{y_{Match}}^+$ ,  $\eta_{\theta_{Match}}^+ \in [1,2]$ ;  $\eta_{x_{Match}}^-$ ,  $\eta_{y_{Match}}^-$  and  $\eta_{\theta_{Match}}^- \in ]0,1[$ ; and finally  $\lambda_{x_{Match}}$ ,  $\lambda_{y_{Match}}$ ,  $\lambda_{\theta_{Match}} \in [0.01,0.1]$ . The best performance, i.e. a lower number of iterations needed for the 3D Perfect Match find a close solution from the true one, in the large majority of the times, was achieved to  $\eta_{x_{Match}}^+ = \eta_{y_{Match}}^+ = \eta_{\theta_{Match}}^+ = \eta^+ = 1.2$ ,  $\eta_{x_{Match}}^- = \eta_{y_{Match}}^- = \eta_{\theta_{Match}}^- = \eta^- = 0.5$  and  $\lambda_{x_{Match}} = \lambda_{y_{Match}} = 0.01$  metres and  $\lambda_{\theta_{Match}} = 0.05$  radians.

It is considered that, a close solution from the true state of the vehicle is obtained when, that solution has an distance error smaller than 4 centimetres, which is the resolution of a cell of the distance and gradient matrices. To be a close solution, also it is needed, that the angle error between the true and the estimated solution is smaller than 0.04 radians.

In this experiment, the initial vehicle pose starts with a distance error equal to eight times the resolution of a cell (32 centimetres) and an angle error of 90 degrees. For the large majority of the cases, the estimated position reached the close solution in less than 10 iterations. On the contrary, in the few cases where the solution not reached the close pose, it was sufficient close to be achieved in the following cycle. Thus, the number of maximum iterations, *iterMax* represented in the Algorithm 9.3, was ten iterations.

Therefore, with this configuration for its parameters, the RPROP algorithm, is able to find a close solution in 10 iterations, with an initial distance error smaller or equal than eight times the resolution of a cell and an initial angle error smaller or equal than 90 degrees.

The number of ten iterations in the RPROP routine guaranteed a maximum execution time of 17 milliseconds, to obtain the vehicle position, using the robot platform (Mini-ITX pc on board), described in the Chapter 5, with 682 points of the tilting Laser Range Finder. The results related with this execution time are presented on the experimental results chapter.

To completely perform the 3D Perfect Match stage, it is necessary to calculate the second derivative. The analysis of the second derivative allows finding an error classification for the 3D Perfect Match solution.

For an actual set of LRF data, if the cost function  $E$  has a perfect distinctive minimum, the second derivative  $\frac{\partial^2 E}{\partial x_{Match}^2}$  is higher. On the contrary, when for those points the cost function  $E$  is flat, and there are not a distinctive minimum, the second derivative is low. Therefore, a higher second derivative represents situations where the confidence in the solution obtained by the 3D Perfect Match algorithm should be higher. In the other hand, a lower second derivative of the cost function, represent cases where the confidence is lower.



Therefore, the 3D Perfect Match covariance matrix represent the error, i.e. the confidence, that there are in the solution reached by the 3D Perfect Match algorithm. This covariance is given by the following matrix:

$$P_{Match} = \begin{bmatrix} \frac{K_{XY}}{\left(\frac{\partial^2 E}{\partial x_{Match}^2}\right)} & 0 & 0 \\ 0 & \frac{K_{XY}}{\left(\frac{\partial^2 E}{\partial y_{Match}^2}\right)} & 0 \\ 0 & 0 & \frac{K_{\theta}}{\left(\frac{\partial^2 E}{\partial \theta_{Match}^2}\right)} \end{bmatrix} \quad (9.28)$$

As can be conclude with the expression (9.28), an higher second derivative decreases the variance value and then increase the confidence in the solution obtained by the 3D Perfect Match when estimating  $x_{Match}$ ,  $y_{Match}$  and  $\theta_{Match}$ . On the contrary, a lower second derivative, increase the variance, decreasing the confidence in the estimated variables.

The parameters of the covariance matrix  $K_{XY}$  and  $K_{\theta}$  are normalization values achieved empirically. The algorithm was tested with different values of  $K_{XY}$  and  $K_{\theta}$  in the gap of  $[1e-6, 1e-1]$ . The best performance was achieved by the values:  $K_{XY} = 1e-3$  and  $K_{\theta} = 1e-3$ . Here, the best performance was found, adjusting the parameters  $K_{XY}$  and  $K_{\theta}$ , making the 3D Matching algorithm smother as possible, when it is performed the fusion, in the Update stage, between this 3D Perfect Match solution and the state obtained during the Prediction stage.

We shall now consider that in the *localization* procedure, the vehicle moves at a maximum speed of 0.4 m/s. With a time cycle of 100 milliseconds, the vehicle will move a maximum distance  $maxDist$ , between time cycles, equal to  $0.4 \times 0.1 = 0.04 metres$ . In this situation, smother as possible means a localization without a variation of the vehicle position lower than two times the value of  $maxDist$ , which is equal to  $0.08 metres$ .

To compute the second derivative, the previously defined cost function, (9.13), is replaced by the quadratic cost function (9.29). This occurs in order to ascertain which cost function is positively definite for all laser scan points. The cost function is given by:

$$E_i = \frac{1}{2} \cdot \left(\frac{d_i}{L_c}\right)^2 \quad (9.29)$$

The second derivatives of the total cost function are given by the vector:

$$\begin{aligned} \nabla E^2 &= [\nabla E_x^2 \quad \nabla E_y^2 \quad \nabla E_{\theta}^2]^T \Leftrightarrow \\ \nabla E^2 &= \begin{bmatrix} \frac{\partial^2 E}{\partial x_{Match}^2} & \frac{\partial^2 E}{\partial y_{Match}^2} & \frac{\partial^2 E}{\partial \theta_{Match}^2} \end{bmatrix}^T \Leftrightarrow \end{aligned}$$

$$\begin{aligned}
\nabla E^2 &= \left[ \sum_{i=1}^N \frac{\partial^2 E_i}{\partial x_{Match}^2} \quad \sum_{i=1}^N \frac{\partial^2 E_i}{\partial y_{Match}^2} \quad \sum_{i=1}^N \frac{\partial^2 E_i}{\partial \theta_{Match}^2} \right]^T \Leftrightarrow \\
\nabla E^2 &= \frac{1}{L_c^2} \sum_{i=1}^N \left[ \frac{\partial^2 d_i}{\partial x_{Match}^2} \quad \frac{\partial^2 d_i}{\partial y_{Match}^2} \quad \frac{\partial^2 d_i}{\partial \theta_{Match}^2} \right]^T \Leftrightarrow \\
\nabla E^2 &= \frac{1}{L_c^2} \sum_{i=1}^N \frac{\partial^2 d_i}{\partial X_{Match}^2}
\end{aligned} \tag{9.30}$$

The second partial derivative vector  $\frac{\partial^2 d_i}{\partial X_{Match}^2}$  of the cost function for  $x_{Match}$ ,  $y_{Match}$  and  $\theta_{Match}$ , relatively to point  $i$ , is given by:

$$\frac{\partial^2 d_i}{\partial X_{Match}^2} = \left[ \frac{\partial^2 d_i}{\partial x_{Match}^2}, \frac{\partial^2 d_i}{\partial y_{Match}^2}, \left( \frac{\partial d_i}{\partial \theta_{Match}} \right)^2 + d_i \cdot \frac{\partial^2 d_i}{\partial \theta_{Match}^2} \right] \tag{9.31}$$

in which  $\frac{\partial d_i}{\partial \theta_{Match}}$  is shown in equation (9.22), while the other derivatives are given by:

$$\frac{\partial^2 d_i}{\partial x_{Match}^2} = \nabla x^2(x_i^W, y_i^W, z_i^W) \tag{9.32}$$

$$\frac{\partial^2 d_i}{\partial y_{Match}^2} = \nabla y^2(x_i^W, y_i^W, z_i^W) \tag{9.33}$$

$$\frac{\partial^2 d_i}{\partial \theta_{Match}^2} = \begin{bmatrix} \nabla x(x_i^W, y_i^W, z_i^W) \\ \nabla y(x_i^W, y_i^W, z_i^W) \end{bmatrix}^T \begin{bmatrix} -\cos \theta_{Match} & \sin \theta_{Match} \\ -\sin \theta_{Match} & -\cos \theta_{Match} \end{bmatrix} \begin{bmatrix} x_i^W \\ y_i^W \end{bmatrix} \tag{9.34}$$

Finally the covariance matrix,  $P_{Match}$ , can be re-written as the following matrix:

$$P_{Match} = \frac{1}{L_c^2} \sum_{i=1}^N \begin{bmatrix} \frac{K_{XY}}{\nabla x^2(x_i^W, y_i^W, z_i^W)} & 0 & 0 \\ 0 & \frac{K_{XY}}{\nabla y^2(x_i^W, y_i^W, z_i^W)} & 0 \\ 0 & 0 & \frac{K_\theta}{\left( \frac{\partial d_i}{\partial \theta_{Match}} \right)^2 + d_i \cdot \frac{\partial^2 d_i}{\partial \theta_{Match}^2}} \end{bmatrix} \tag{9.35}$$

### 9.1.3. Kalman Filter Update

The Kalman Filter Update stage combines the estimated state using the odometry and the 3D Perfect Match procedure,  $\hat{X}_v(k+1|k)$  and  $X_{Match}$ , respectively. The Kalman Filter equations can be seen in more detail in [53].

The observation model  $h_v$  in the update stage is equal to the vehicle state  $X_v$ :

$$h_v = [x_v \quad y_v \quad \theta_v]^T \tag{9.36}$$

Therefore, the estimated observation is equal to the present estimation of the vehicle state, propagated during the Kalman Filter Prediction stage,  $\hat{X}_v(k+1|k)$ :

$$\begin{aligned}\hat{h}_v &= [\hat{x}_v \quad \hat{y}_v \quad \hat{\theta}_v]^T(k+1|k) \Leftrightarrow \\ \hat{h}_v &= \hat{X}_v(k+1|k)\end{aligned}\quad (9.37)$$

Furthermore, the observation of the vehicle state can be written as the following expression:

$$Z_v = h_v + r \quad (9.38)$$

in which  $r$  is white noise, modelled with a Gaussian distribution with zero mean ( $\hat{r} = 0$ ) and covariance matrix  $R$ . Therefore, in the update stage the observation is equal to the state obtained after the application of the 3D Perfect Match:

$$Z_v = X_{Match} \quad (9.39)$$

Hence, the innovation of the Kalman Filter  $V(k+1)$  is equal to:

$$\begin{aligned}V(k+1) &= Z_v - \hat{h}_v \Leftrightarrow \\ V(k+1) &= X_{Match} - \hat{X}_v(k+1|k)\end{aligned}\quad (9.40)$$

In this stage, the propagated covariance matrix, using odometry  $P_v(k+1|k)$  and the covariance matrix provided by the 3D Perfect Match procedure  $P_{Match}$ , are used to determine the Kalman Filter Gain  $W(k+1)$ .

As the observation  $Z_v$  is equal to  $X_{Match}$ , the error covariance matrix  $R$ , is equal to the covariance matrix obtained during the 3D Perfect Match,  $P_{Match}$ . Therefore, the Kalman Filter Gain is given by the following expression:

$$W(k+1) = P_v(k+1|k) \frac{\partial h_v^T}{\partial X_v} \left[ \frac{\partial h_v}{\partial X_v} P_v(k+1|k) \frac{\partial h_v^T}{\partial X_v} + \frac{\partial h_v}{\partial r} P_{Match} \frac{\partial h_v^T}{\partial r} \right]^{-1} \quad (9.41)$$

The gradient of the observation model, equation (9.36), in order to the vehicle state  $X_v$  and the observation noise  $r$ ,  $\frac{\partial h_v}{\partial X_v}$  and  $\frac{\partial h_v}{\partial r}$  respectively, are identity matrices. Therefore, the previous equation can be re-written as follows:

$$W(k+1) = P_v(k+1|k) [P_v(k+1|k) + P_{Match}]^{-1} \quad (9.42)$$

Therefore, after the update stage the new estimated state is given by the expression:

$$\hat{X}_v(k+1|k+1) = \hat{X}_v(k+1|k) + W(k+1)V(k+1) \quad (9.43)$$

The new covariance matrix, decreases with the following equation:

$$P_v(k+1|k+1) = [I^{3 \times 3} - W(k+1)]P_v(k+1|k) \quad (9.44)$$

where  $I^{3 \times 3}$  is the square identity matrix with the dimension  $3 \times 3$ .

The Kalman Filter Update stage can be summarized with the following equations.

Algorithm 9.5 Equations of the Kalman Filter Update stage.

<b><i>KalmanUpdate</i></b> ( $\hat{X}_v(k+1 k), P_v(k+1 k), X_{Match}, P_{Match}$ )
$W(k+1) \leftarrow P_v(k+1 k)[P_v(k+1 k) + P_{Match}]^{-1},$ $V(k+1) = X_{Match} - \hat{X}_v(k+1 k)$ $\hat{X}_v(k+1 k+1) = \hat{X}_v(k+1 k) + W(k+1)V(k+1)$ $P_v(k+1 k+1) = [I^{3 \times 3} - W(k+1)]P_v(k+1 k)$

# 10. Experimental Results

The final methodology for the *three-dimensional map-based* approach is composed of: 1) *pre-localization* and *mapping*; and finally 2) *localization*. About this approach three papers were published [70] to [72].

The results presented in this chapter are about the application of the entire methodology (*pre-localization* and *mapping*, *localization*) in eight different indoor scenarios. Some videos about the tested entire methodology, inside these scenarios can be accessed in the webpage of the PhD thesis presented in this document, [3]. Press's videos and news about these demonstrations can also be find in this webpage.

Three of these scenarios were inside the Faculty of Engineering of the University of Porto (FEUP). The RobVigil project was also demonstrated in public facilities. These public demonstrations are the others five environments where the entire methodology was tested. These eight scenarios can be described as follows: 1) the floor -1 of the Department of Electrical and Computers Engineering of the FEUP, known as building I; 2) the faculty teacher's garage; 3) the main corridor of the FEUP, known as building B; 4) the multipurpose pavilion of Guimarães ("pavilhão multiusos de Guimarães"), during the competition of the FreeBots, in the Portuguese National Festival of Robotics, the "Robótica 2012", [73], which took place between 11th and 15th of April; 5) a corridor of the INESC-TEC building in the first floor, during the UTM unit demonstration, in 3 of April of 2012, [74]; 6) "Fórum do Mar 2013" fair, which happened in the Exponor, Porto, during May of 2012, in the days 10,11 and 12, [75]; 7) the shopping gallery of the enterprise centre "Lionesa" in Matosinhos, during the day of 5 of July; and finally 8) the lobby space at the "reitoria" of the University of Porto, at 28 of September of 2012, during the night of the Europeans researchers.

## 10.1. How to Setup the Localization for Operation in a New Scenario

Imagine that the vehicle will operate in a new indoor scenario and it is necessary to setup the localization for its correct operation. This sub-section will describe how this setup is made.

For the setup of the localization for a new scenario, it is necessary follow the next guide, presented in the following table, Table 10.1.

In the pre-localization the used values for  $Z_{min}$  and  $Z_{max}$  (minimum and maximum z coordinates), as defined in Chapter 3 and Chapter 7, were 0.2 and 1.8 metres, respectively.

The angle  $\beta_{min}$  represented in the following figure, Fig. 10.1 is the minimum rotation of the tilting LRF during the EKF-Loc phase. Since during this phase, only linear segments with a minimum size of 0.7 metres are used, the projection in the horizontal of any linear segment, should have a minimum size of 0.7 metres. As the vehicle height is 1.3 metres,  $\beta_{min}$  is equal to the following expression:

$$\beta_{\min} = -\tan^{-1}\left(\frac{1.3}{0.7}\right) \approx -62^\circ \quad (10.1)$$

Stage	Phase	Description
1) Pre-Localization	EKF-SLAM	-Controlling the vehicle with joystick, a log of data (odometry and LRF points) is saved with the vehicle moving at 0.2 m/s and the observation module fixed in the horizontal.  -Offline, the EKF-SLAM runs and is obtained a 2D feature map of linear segments and invariant points.
	EKF-Loc	-Controlling the vehicle with joystick a log of data (odometry and LRF points) is saved with the vehicle moving at 0.2 m/s and Laser rotating at 10 rpm, between of $\beta_{\min}$ and $\beta_{\max}$ .  -Offline, the EKF-Loc runs and it is obtained the vehicle pose.
2) Mapping	Bayes Rules	The same log, used in the previous phase, together with the obtained vehicle pose, allows obtaining the occupancy grid of the environment, in the three-dimensional space.
	Distance Transform	Using the final occupancy grid and the distance transform, it is obtained the 3D distance matrix.
	Sobel Filter	Using the Sobel Filters over the distance matrix it is possible to create the 3D gradient matrices in order to x and y directions.
3) Localization	Normal Operation	The stored look-up tables (distance and gradient matrices in the three-dimensional space) are used during the 3D matching localization algorithm.  The vehicle navigates autonomously between goal points, with an average speed of 0.4 m/s, with the laser rotating at 10 rpm, between of $\gamma_{\min}$ and $\gamma_{\max}$ , using only the static scenario (upper side of a building- above 1.8 metres ) to pinpoint itself position.

Table 10.1 How to setup the Localization for a new scenario. Brief description of each phase.

It is intended, during the EKF-Loc phase, to have data about the upper side of a building. Therefore the angle  $\beta_{\max}$  should be  $90^\circ$ .

The angles  $\gamma_{\min}$  and  $\gamma_{\max}$  are represented in the following figure, Fig. 10.2. The value of  $\gamma_{\min}$  represents the minimum angle of the tilting LRF during the *localization*. Since, only data of the headroom is used (above 1.8 metres of height),  $\gamma_{\min}$  is computed in the following way:

$$\gamma_{\min} = \sin^{-1}\left(\frac{1.8 - 1.3}{LRF_{Range}}\right) \quad (10.2)$$

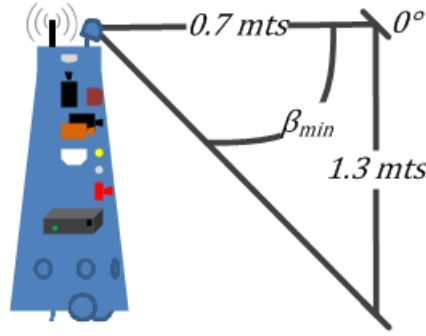


Fig. 10.1 Linear segment projection on the horizontal need to have at least 0.7 metres.  $\beta_{min}$  follows this rule.

The laser range finder range is  $LRF_{Range}$ , equal to 5 metres. Therefore,  $\gamma_{min}$  can be computed as follows:

$$\gamma_{min} = \sin^{-1}\left(\frac{0.5}{5}\right) \approx 6^\circ \quad (10.3)$$

Finally, the angle  $\gamma_{max}$  depends if the information of the ceiling is or not helpful for the 3D matching algorithm. If the ceiling has helpful information,  $\gamma_{max}$  should be  $90^\circ$ . On the contrary  $\gamma_{max}$  should be computed as the following equation:

$$\gamma_{max} = \sin^{-1}\left(\frac{H_{building} - 1.3}{LRF_{Range}}\right) \quad (10.4)$$

in which,  $H_{building}$  represents the building height.

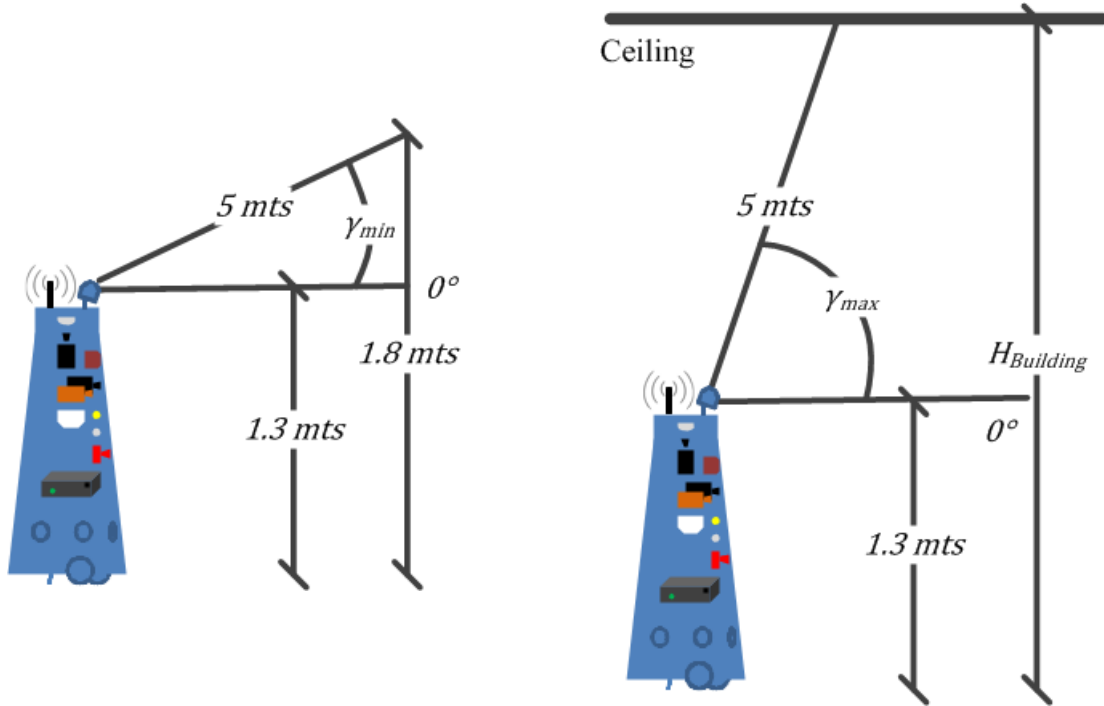


Fig. 10.2  $\gamma_{min}$  and  $\gamma_{max}$  representation.

## 10.2. Pre-Localization Results

The *pre-localization* is a computationally time consuming task. However, as this procedure is performed only once and offline, the computational time is not a problem.

If the vehicle pose is known while the observation module is acquiring data about the surrounding environment, it is possible to build the occupancy grid, the distance and gradient matrices, as described in the Chapter of *Mapping* (8), which are used during the vehicle normal operation, as described in Chapter of *Localization* (9).

In the *pre-localization* procedure the EKF-SLAM stage becomes possible to obtain the feature map, with linear segments and invariant points, about the environment which is intended to map. The use of this feature map, allows applying the EKF-Loc stage.

In this sub-section, several feature maps about the eight scenarios described above are shown. Together with these feature maps are shown the architectural plan about the building. As it is possible to see in some situations the architectural plan is obsolete or it does not capture the entire characteristics and details when comparing with the feature map.

The following figure, Fig. 10.3, shows the architectural 2D plan of the scenario 1) and Fig. 10.4 shows feature map obtained with the application of the EKF-SLAM. This scenario, is inside the Department of Electrical and Computers Engineering of FEUP and is constituted with a corridors, hallways and a laboratory, occupying a total of  $60 \times 60 \text{ meters}^2$ .

The figure of the feature map, Fig. 10.4, shows the mapped invariant points in small red points. The mapped linear segments are represented in red linear segments. The black and continuous line represents the vehicle trajectory applying the EKF-SLAM cycle. In blue and dotted line, it is possible to see the trajectory of the vehicle using odometry only. Through the comparison of the trajectories can be seen that the localization achieved with odometry is completely inconsistent, while the vehicle position computed during the EKF-SLAM procedure is correct.



Fig. 10.3 Architectural plan of the scenario 1). The solid black shows the zone where the *pre-localization* was applied. 1,2,3 and 4 are the divisions of the map.



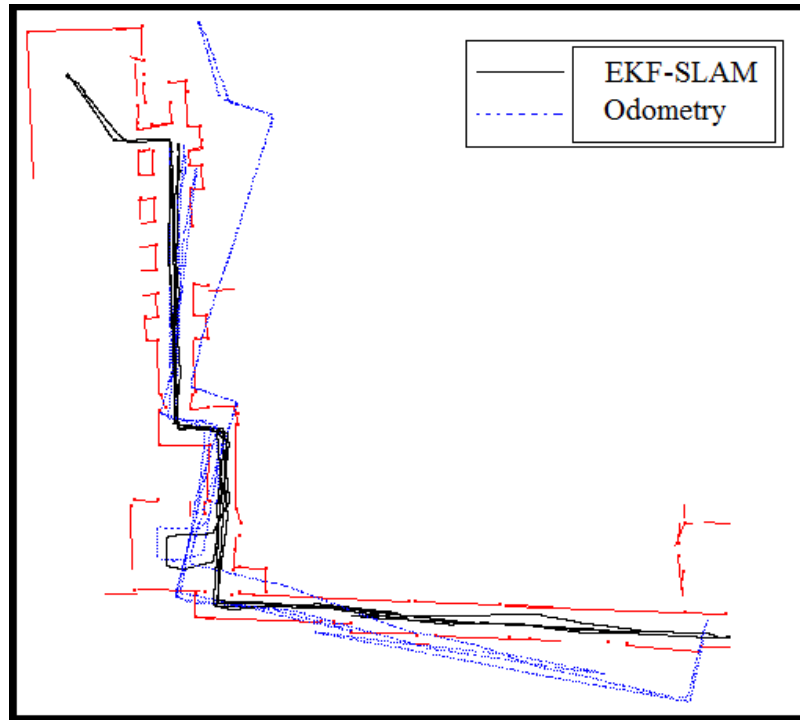


Fig. 10.4 2D map obtained with the application of the EKF-SLAM procedure. A square area of 60 metres x 60 metres.

The two trajectories shown at Fig. 10.5 are about the same data, used during the EKF-SLAM and the EKF-Loc stages. The black and continuous line represents the vehicle trajectory in the EKF-SLAM cycle. In blue and dotted line, it is possible to see the trajectory of the vehicle in the EKF-Loc using the feature map previously built by the EKF-SLAM filter.

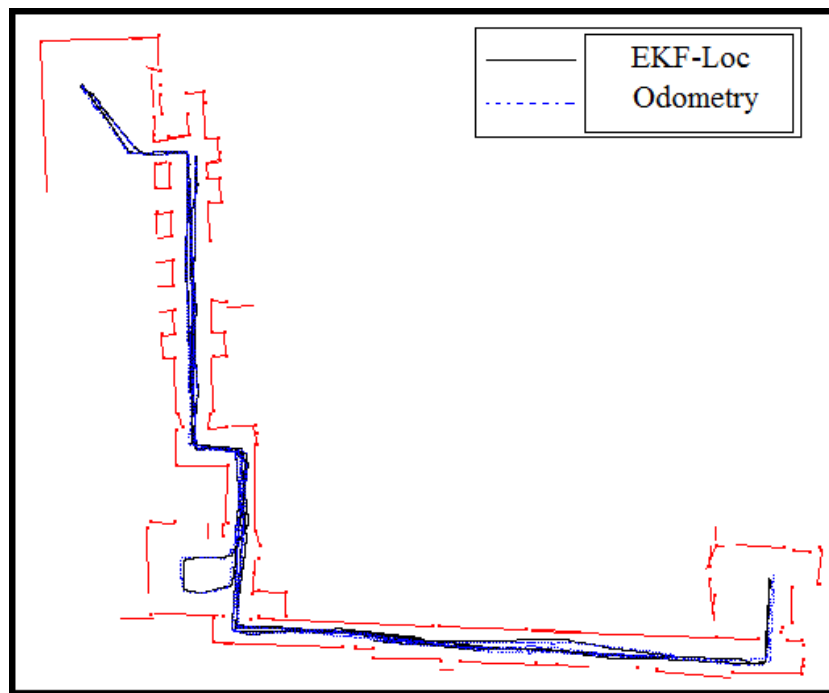


Fig. 10.5 EKF-SLAM procedure trajectory Vs EKF-Loc trajectory.

The difference between the trajectories happens due the continuous map's correction, in the EKF-SLAM cycle, which happens in the map vector, when there are a closure loop. When applied the EKF-Loc, the map remains static, therefore it produces a different trajectory. This continuous corrections makes necessary to apply the EKF-Loc after obtained the feature map of the EKF-SLAM, aiming a more accuracy in the estimation of the vehicle pose.

The map of Fig. 10.4 is constituted with 219 features, 117 invariant points and 102 linear segments. The map was obtained with a total distance travelled of 367 metres . A map with this dimension introduces a lot of features in the Extended Kalman Filter, which became the EKF-SLAM impracticable to operate online. In this case, at each iteration, with 219 features, the EKF-SLAM procedure spends about 3 seconds in an Laptop Intel Core 2 Duo 2.53GHZ.

In fact, this is not a problem, since the *pre-localization* is performed only once and offline, as preparation stage to obtain the distance and gradients matrices used during the *localization*. Although, the absence of restrictions in terms of spent time during the *pre-localization* and *mapping* procedures, when the entire map is divided, into smaller maps, the number of mapped features decreases, reducing the computational time spent. Besides that, when the EKF-SLAM procedure is applied in smaller maps, the vehicle travelled distance decreases, allowing an easier loop closure, increasing the accuracy of the final feature map.

Thus, the map of Fig. 10.3, can be divided in smaller maps, as shown in numbers 1,2,3 and 4. After applying the EKF-SLAM to each zone, allowing to obtain the respective feature maps, shown in figures: Fig. 10.6 to Fig. 10.9. The mapping procedure can be applied individually to each smaller map.

A particular hard map to be mapped is the corridor represented at Fig. 10.3 with the number 4. This corridor has little information along its translational direction. The tilting LRF has a small range and therefore, it is not able to "see" the start and the end of the corridor every time. Furthermore, this corridor has little information about the connectivity to the rest of the map. There are a small passage between the maps marked with the number 3 and 4 in the Fig. 10.3, which becomes hard, to represent correctly, the relation between them. The relation between these two maps, 3 and 4 , should be orthogonal. As it is possible to see in Fig. 10.4, the resultant relation is lightly different of 90 degrees.

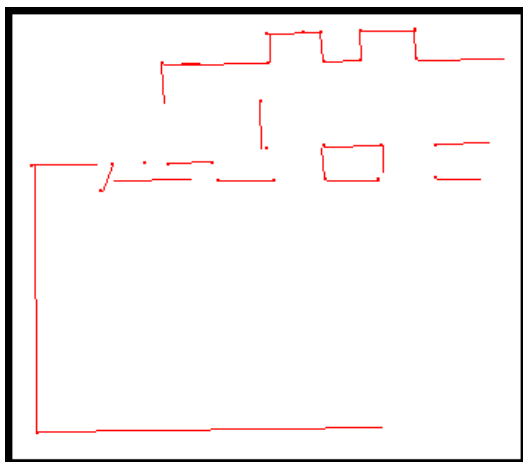


Fig. 10.6 Map 1. Features:52; points:27 and linear segments:25. 25 metres x 25 metres of square area.

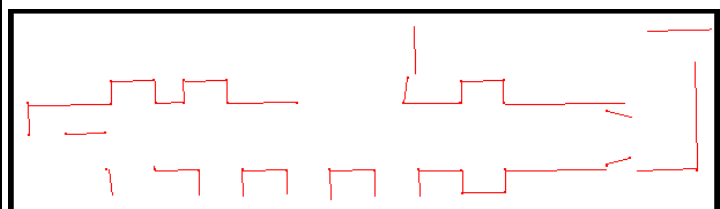


Fig. 10.7 Map 2. Features:74; points:35 and linear segments:39. Corridor with 25 metres length.

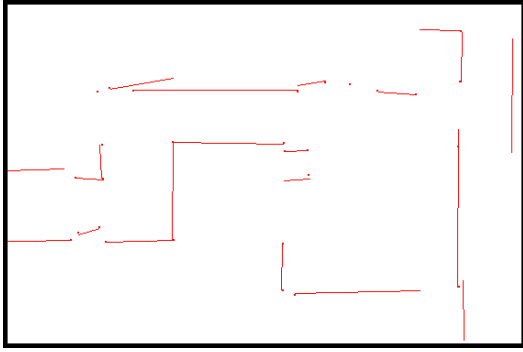


Fig. 10.8 Map 3. Features:50; points:28 and linear segments:22. Area of  $25 \times 25 \text{ metres}^2$ .

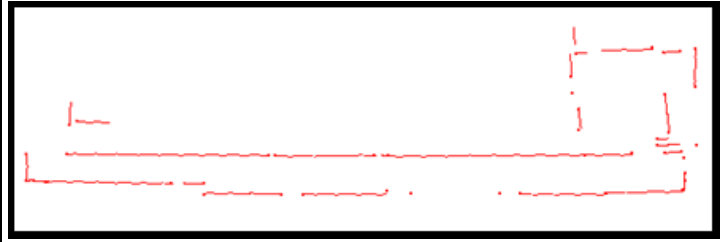


Fig. 10.9 Map 4. Features:75; points:44 and linear segments:31. Corridor with about 50 metres.

Therefore, applying the *pre-localization* in smaller maps, as is the case of this corridor, can produce more accurate results. The feature map, in the Fig. 10.9, about this corridor is notoriously more accurate, when compared with the same corridor in the Fig. 10.4. At the final, the 3D occupancy grids obtained individually for each of these maps, can be merged since the relative position between the 3D occupancy grids are well known.

In the following figures, Fig. 10.10, Fig. 10.11 and Fig. 10.12, are shown parts of the 2D map obtained after the EKF-SLAM procedure application, in the buildings defined as scenarios 2) and 3). The entire occupancy grid about these scenarios, 2) and 3), are shown in the following sub-section, the Fig. 10.28 and Fig. 10.30.

In the particular case of the scenario 2), the features represented in the map of Fig. 10.10, are composed essentially by invariant points about circular columns. Also, this scenario is particularly hard to map, since there are less information to be used.

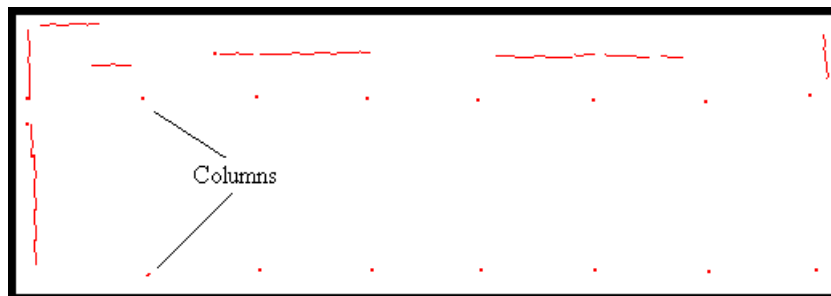


Fig. 10.10 A part of the scenario defined as scenario 2). Features:33, points:19, linear segments:14. Area of 20 metres per 60 metres.

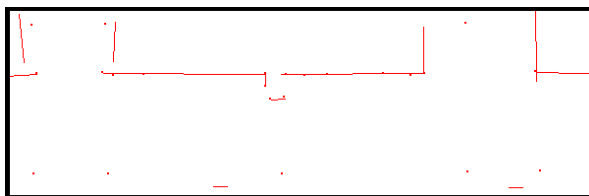


Fig. 10.11 A part of the scenario defined as scenario 3). Features:37, points:23, linear segments:14. Corridor with 30 metres of length.

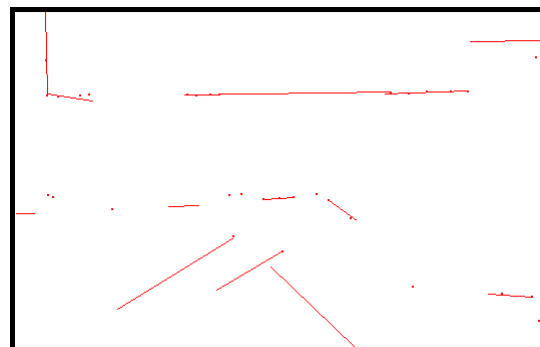


Fig. 10.12 A part of the scenario defined as scenario 3). Features: 53, points: 38, linear segments: 15. Area of 30 metres x 30 metres.

With regard with the scenario 3), it is the main corridor of the FEUP. This corridor has lot of information, walls, columns and corners. Therefore, it proved to be ideal for the application of the *pre-localization* procedure.

Other scenario where this methodology was tested, was in the multipurpose pavilion of Guimarães (“pavilhão multiusos de Guimarães”), scenario referred above as 4). The architectural plan, Fig. 10.13, shows the zone that was mapped, identified with black solid colour. The resultant feature map, obtained after the EKF-SLAM procedure can be seen in figure Fig. 10.14.

This scenario is a corridor with a length of about 60 metres. This corridor was chosen to be used as navigation scenario during the public demonstration, in the Free-Bots competition, which happened in the "Festival Nacional de Robótica", in the city of Guimarães, in 2012.

This corridor has lot of helpful information to be used in the EKF-SLAM filter. It has square columns, which proved to be really helpful, for the building of the feature map.

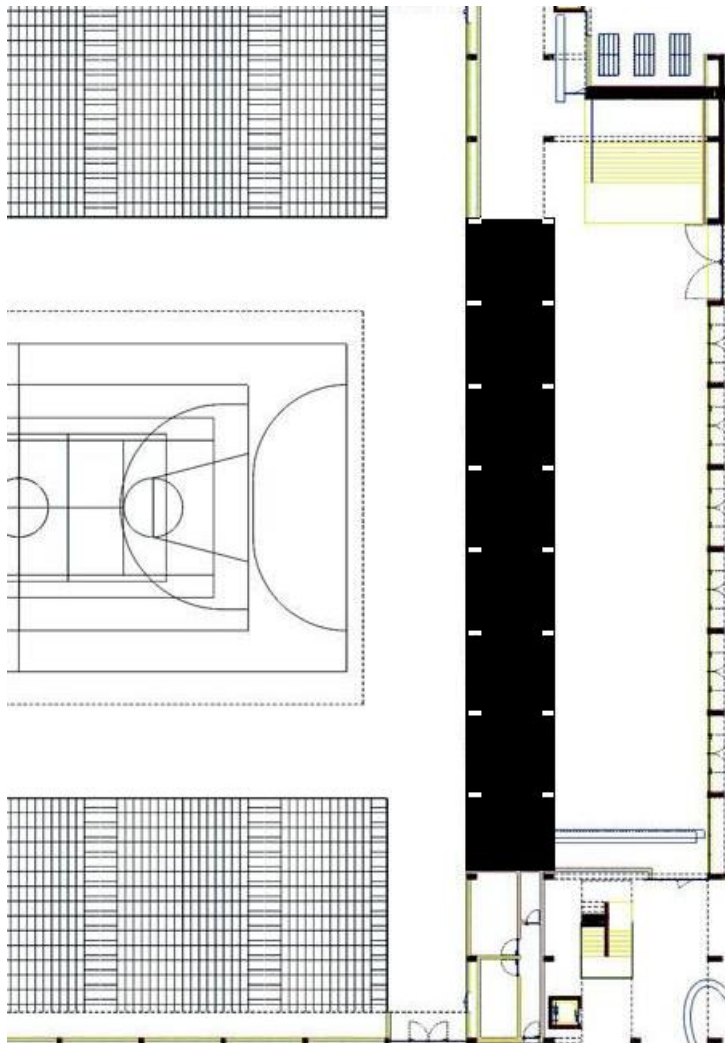


Fig. 10.13 Architectural plan of the (“pavilhão multiusos de Guimarães”), scenario 4). The rectangle in black solid colour, is representative of the zone where the pre-localization was applied.

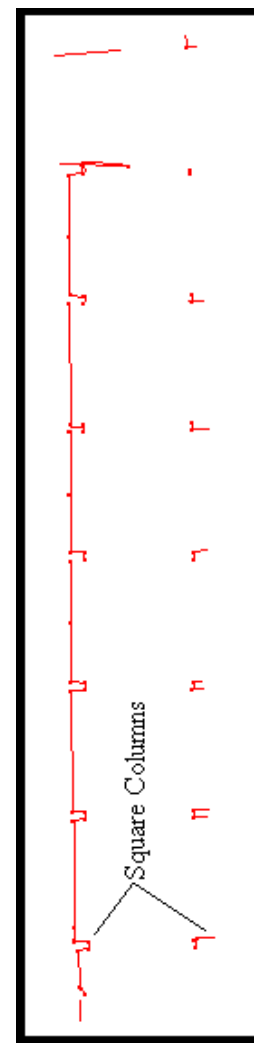


Fig. 10.14 2D map obtained with the application of the EKF-SLAM procedure, scenario 4). Features:120; points:53 and linear segments:67. A corridor with a length of 60 metres.

About the scenario defined above as 5), the Fig. 10.15 shows the building architectural plan. At black solid colour it is represented the mapped zone. The obtained feature map, after the application of the EKF-SLAM, during the *pre*-localization, is shown at Fig. 10.16.

This scenario is the main corridor of the INESC Porto. It is constituted by the building entrance, the elevators access and the reception desk. This corridor has a lightly curvature on its walls, as indicated in the Fig. 10.15. This curvature was successfully modelled in the feature map. Furthermore, this scenario has lot of information, has circular column, corners, walls, entrances and in halls to be used the EKF-SLAM and EKF-Loc stages.

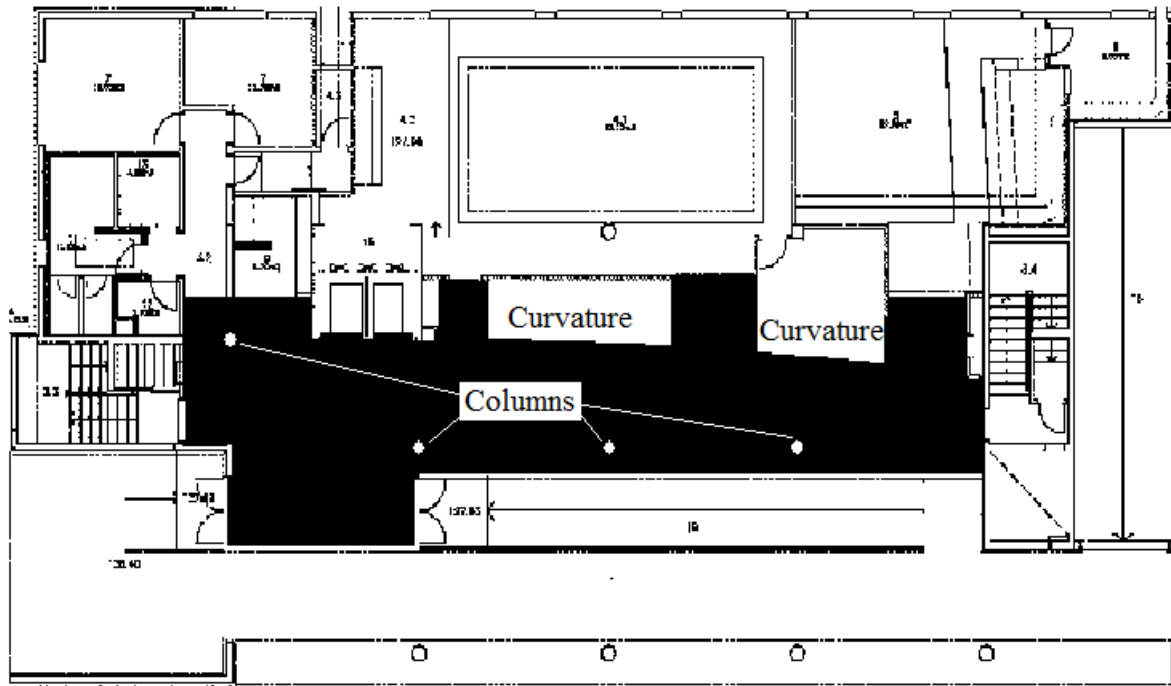


Fig. 10.15 Architectural plan of the INESC TEC building, first floor, scenario 5).

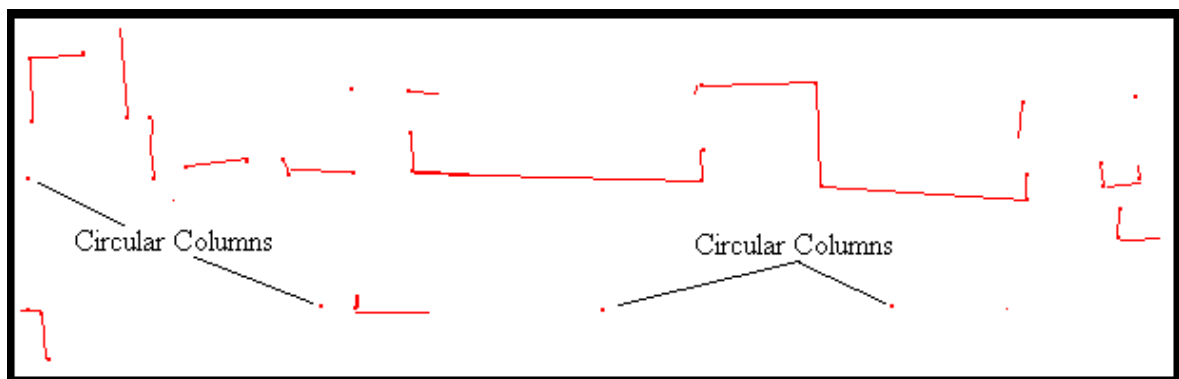


Fig. 10.16 2D map obtained with the application of the EKF-SLAM procedure, scenario 5). Features:81; points:42 and linear segments:39. A corridor with a length of 25 metres.

The following two figures are about the scenario defined as 6). Fig. 10.17 shows the architectural plan about the Exponor building. The zone painted in solid black is the space chosen for the public demonstration. The other figure, Fig. 10.18 shows the result of the application of the EKF-SLAM in this scenario.

In the particular case of this scenario, it was difficult to achieve successful results during the EKF-SLAM. This scenario is really open and has distant features. For example, the distance between the two circular columns, indicated in Fig. 10.18, is higher than ten metres. As the tilting LRF has a small range of 5 metres, when the vehicle is between the columns, there are zones where it is completely "blind", since it is not capable of "see" anyone of the existing columns or other features.

Therefore, during the *pre-localization* procedure, it was necessary to have particularly care with the trajectory performed by the vehicle, aiming its displacement inside areas with reachable and helpful data.

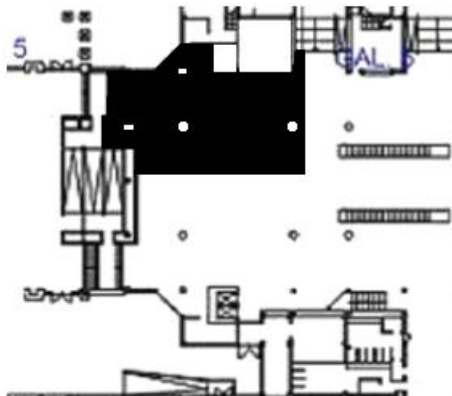


Fig. 10.17 Architectural plan of the mapped part in the "Fórum do Mar" fair, inside the Exponor building (Matosinhos), scenario 6).

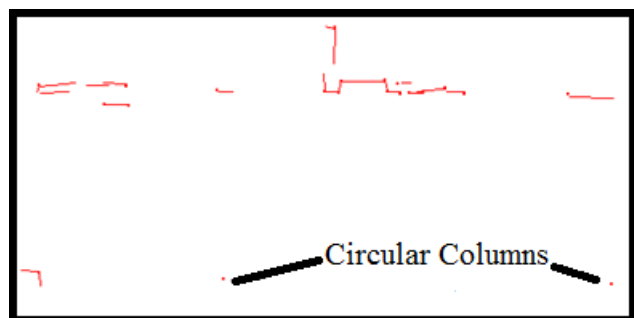


Fig. 10.18 2D map obtained with the application of the EKF-SLAM procedure, scenario 6). Features:45; points:21 and linear segments:24. A corridor with a length of 25 metres..

The following figure, Fig. 10.19, is the architectural plan of the shopping gallery inside the enterprise centre "Lionesa", defined above as the seventh scenario. Particularly this scenario, has lot of people walking. Therefore, the mapping was made during the night. The other figure, Fig. 10.20, shows the obtained feature map, when applied the EKF-SLAM.



Fig. 10.19 Architectural plan of a gallery shopping, inside the enterprise centre "Lionesa" (Porto), scenario 7) The solid black colour was the mapped area.

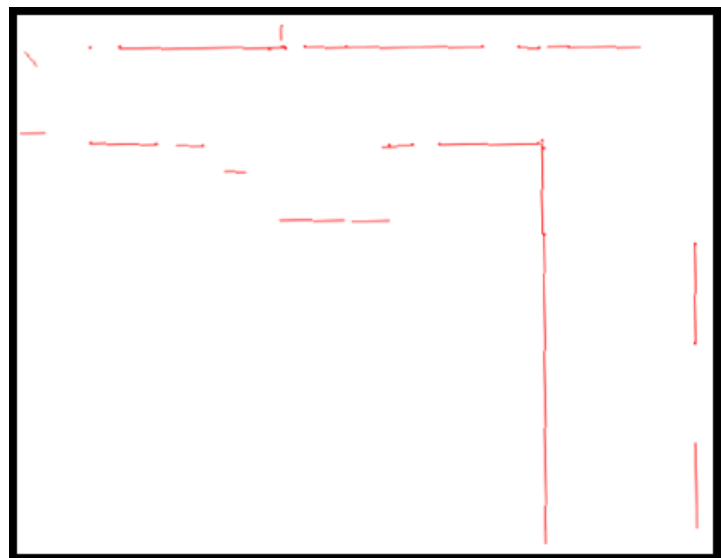


Fig. 10.20 2D map obtained with the application of the EKF-SLAM procedure, scenario 7). Features:42; points:22 and linear segments:20. Square area of 30 metres x 30 metres.

This last scenario is well-structured and also proved to be the ideal kind of scenario to apply the entire mapping methodology. As it can be seen by the architectural plan, the Fig. 10.19, this corridor forms an "L" word. As this scenario is a shopping gallery, it has glass windows/walls, allowing to see the stores inside, which only the feature map has the ability of obtain, as shown in Fig. 10.20. In fact, the architectural plan is not capable of capture the entire detail, that the feature map is.

With respect to the 8th scenario, the reitoria's lobby of the University Porto, the architectural plan is shown at Fig. 10.21. The feature 2D map was obtained with success and is shown at the Fig. 10.22.

This scenario has lot of details, which were really helpful to the EKF-SLAM and EKF-Loc phases. As it is possible to see at Fig. 10.22, the resultant feature map is composed of a huge number of linear segments and invariant points. Again, this feature map is capable of capture details that are not represented in the architectural plan.

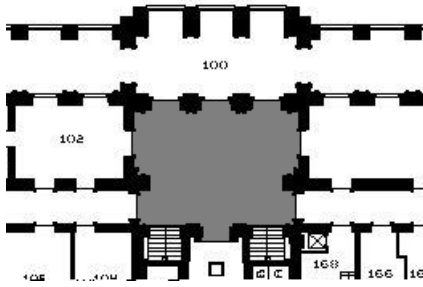


Fig. 10.21 Architectural plan of the reitoria lobby of the University Porto, scenario 8). The solid grey colour was the mapped area.

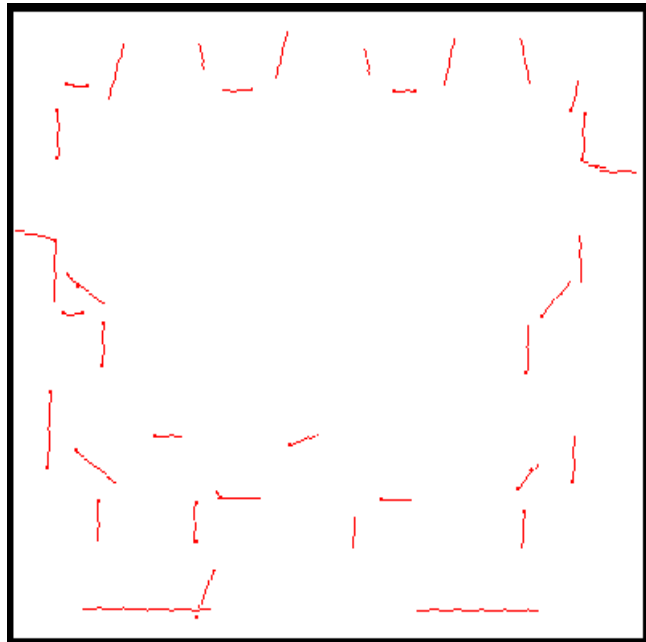


Fig. 10.22 2D map obtained with the application of the EKF-SLAM procedure, scenario 8). Features:70; points:33 and linear segments:37. Square area of 25 metres x 25 metres.

The correctness and accuracy of these feature maps are really important. The EKF-Loc uses this maps to obtain the vehicle pose, which is used for the 3D mapping. Therefore, an higher accuracy of the feature map, will also correspond to an higher accuracy in the 3D occupancy grids. As the 3D map of the building is used in the vehicle normal operation, during the *localization* procedure, to perform the 3D matching algorithm, an higher correctness and accuracy of the 3D occupancy grids will result in an higher accuracy in the estimation of the vehicle pose, during the *localization* procedure.

Therefore, these phases are related in terms of correctness and accuracy. In fact, accurate localization results, will correspond to accurate and successful results achieved during the *pre-localization* and *mapping*. In the sub-section of *localization results* 10.4, it will be proved the accuracy of the obtained feature maps.



### 10.3. Mapping Results

The next figure, Fig. 10.23, shows the mapping performed without any treatment to the observation module's data. The white points in Fig. 10.23 (images on the left and on the right) are the set of points  $P$ , with coordinates  $(x, y, z)$ . In summary, the white points are the raw data  $P$  obtained with the observation module without applying the Bayes rules.

In the image on the left, only odometry is used as a localization method. The mapping process becomes inconsistent as a result of the error which occurs while estimating the position. Therefore the resultant map is a wrong representation of the indoor building that is intended to map.

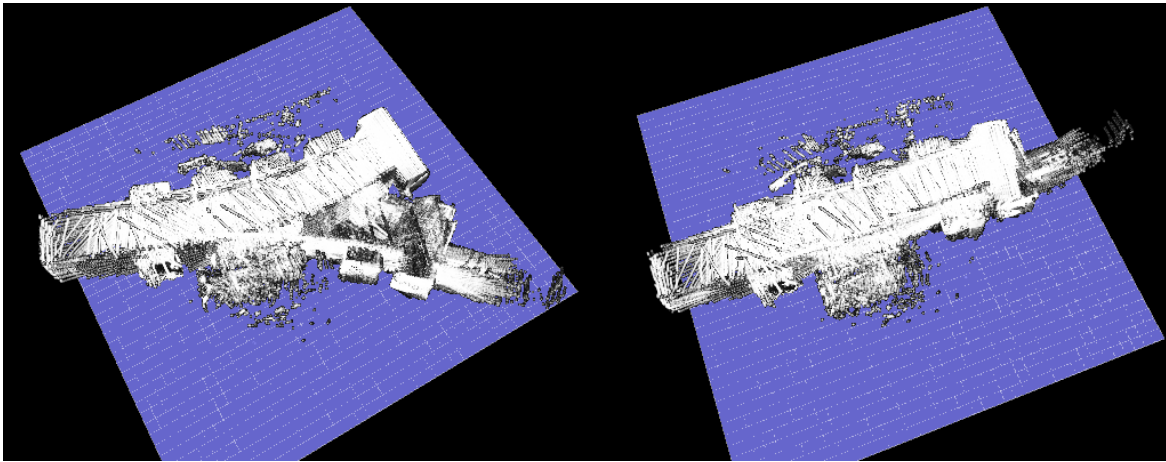


Fig. 10.23 Image on the left: mapping with odometry. Image on the right: mapping with the EKF-Loc stage algorithm (pre-localization procedure).

On the contrary, when the *pre-localization* algorithm is used as method of localization, the mapping is consistent, as shown in Fig. 10.23, image on the right.

The following figure, Fig. 10.24 shows the location where tests were conducted, while Fig. 10.25 is representative of the mapping performed in this scenario, applying the Bayes rules in the raw data  $P$ . The figure is representative of the occupancy grid, with limits equals to:  $x_{max} = 24 \text{ metres}$ ,  $y_{max} = 6 \text{ metres}$  and  $z_{max} = 4 \text{ metres}$ , with a resolution of  $res = 0.04 \text{ metres}$ .



Fig. 10.24 Location where the tests were conducted. Corridor at the Faculty of Engineering of the University of Porto, Portugal.

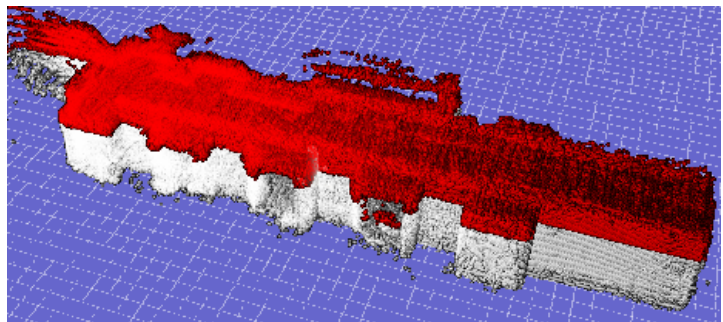


Fig. 10.25 Map grid built using Bayes rules.



Therefore, the cell points printed (in red or white) correspond to the cells where the probability of being occupied is greater than or equal to 0.7. In this figure, the red points correspond to the cells with z coordinates higher than 1.8 metres .

Furthermore, the red points are representative of the zone of the map used during the vehicle normal operation, in the *localization* procedure, considered the upper side of a building, remaining static during large periods of time.

Along this Chapter, the entire set of 3D occupancy grids that are presented will follow the same rule: the red points represent the upper side of the building (with z coordinate above 1.8 metres).

### 10.3.1. 3D Occupancy Grid Maps

As already referred, the entire methodology described in this PhD thesis, was exhaustive tested in the real robot (RobVigil), in eight scenarios. Therefore, the *pre-localization* and *mapping* were performed in the eight different scenarios, allowing the localization and navigation in environments with large indoor sizes.

The 3D occupancy grid about the first scenario is shown in figure Fig. 10.26. The occupancy grid is shown with different perspectives of view.

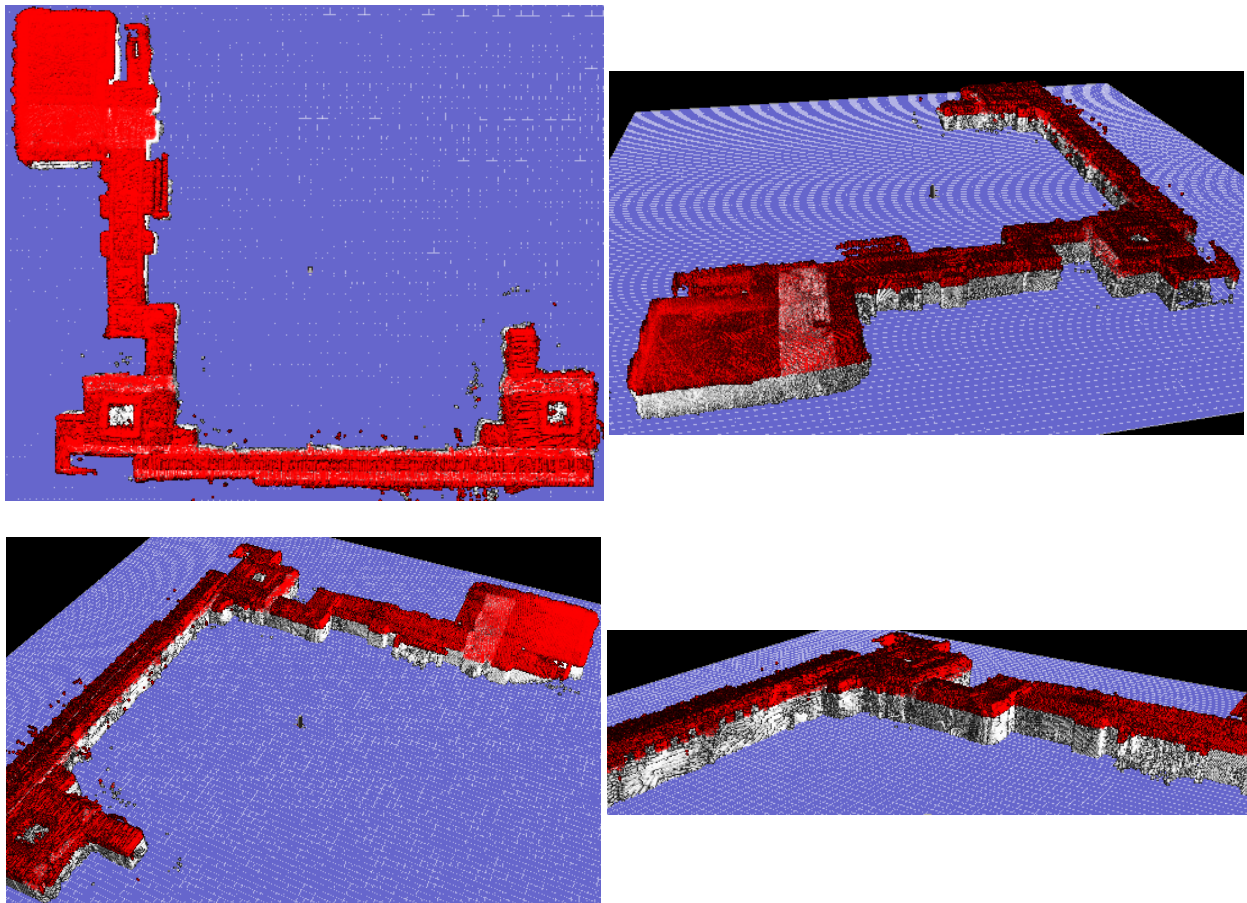


Fig. 10.26 3D Occupancy grid of the scenario defined as 1). Different perspectives of view.

This scenario has a square area of about 60 metres x 60 metres. It is constituted with corridors, one laboratory room, one lobby and two coffee zones. In this scenario there are continuously people travelling and crossing the vehicle navigation area.

In this scenario the ceiling has helpful information to be used when estimating the vehicle pose, using three-dimensional data. Some examples are: air conditioned tubes, hall entrances, light supports, windows *et cetera*.

The following figure, Fig. 10.27, represents the architectural plan about the scenario 2). In this plan, in solid black, is represented the zone where the *pre-localization* and *mapping* was applied. The other figure, Fig. 10.28, shows the obtained occupancy grid, with different perspectives of view.

The area of this scenario is about 60 metres x 60 metres. It is an open scenario, constituted essentially by circular columns with a diameter of about 0.3 metres. It is in continuous changing, with cars entering and leaving the garage. The ceiling of this garage is composed of a matrix with cube shape, as it is possible to see in the occupancy grids represented below. This matrix is helpful information for the vehicle location, using the 3D matching algorithm in the *localization* procedure.

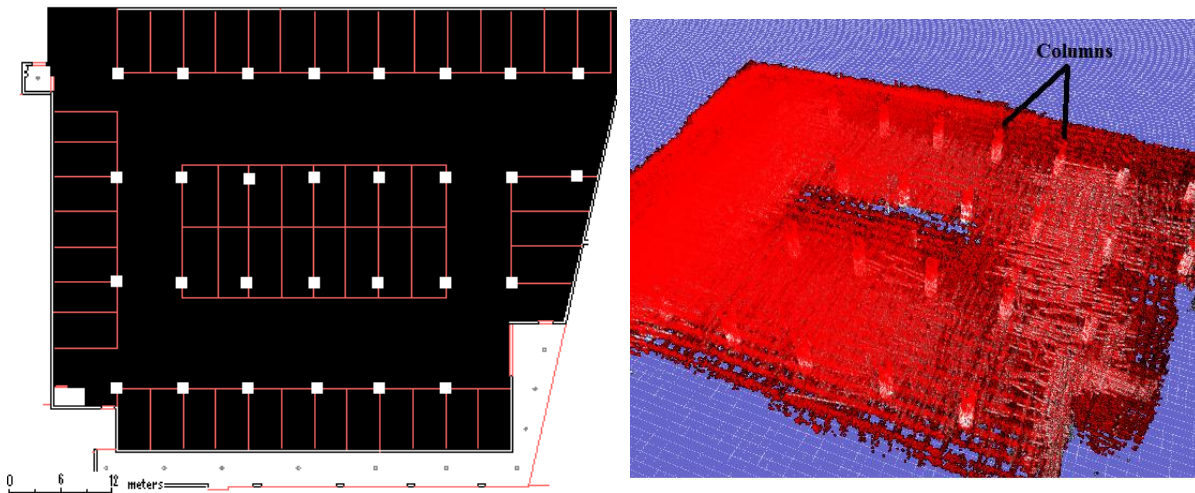


Fig. 10.27 Architectural plan of the scenario defined as 2) Feup teachers' garage. The black solid colour represents the zones which had been mapped.

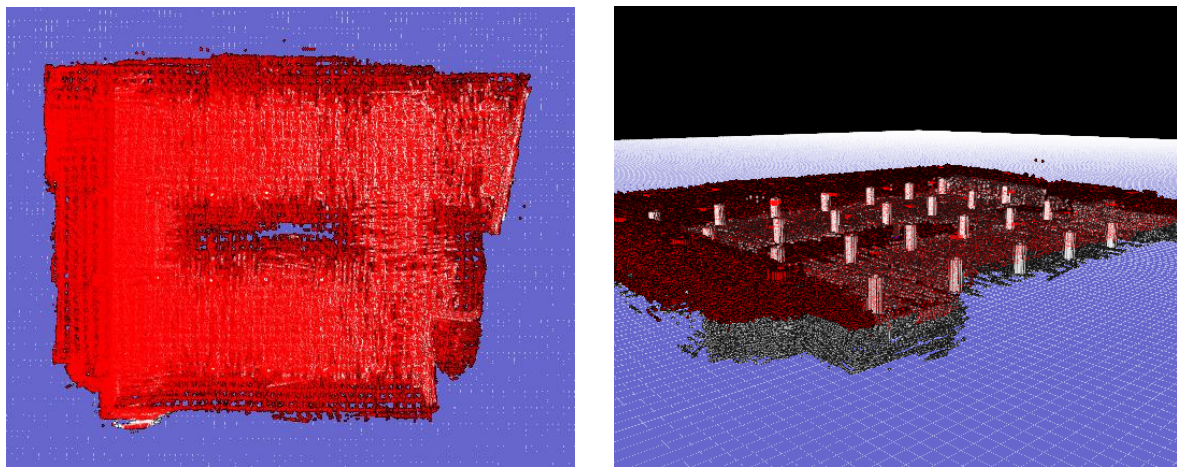


Fig. 10.28 3D Occupancy grid of the scenario defined as 2). With circular columns. Different perspectives of view.



The third scenario is represented with the following figures, Fig. 10.29 and Fig. 10.30. The Fig. 10.29, shows the 2D map about the scenario 3). In solid black, are the zones where the *pre-localization* and *mapping* were applied. The occupancy grids shown in the Fig. 10.30, are about the same scenario, but represented in different perspectives.

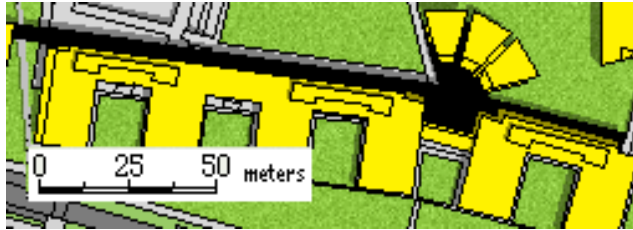


Fig. 10.29 2D map of the scenario defined as 3) students' corridor B. The black solid colour represents the zones which had been mapped.

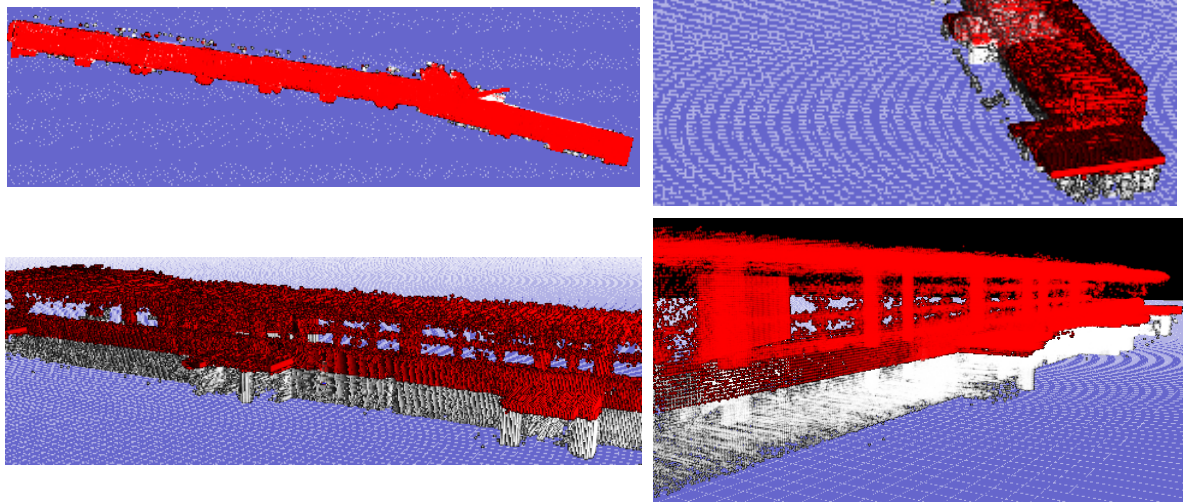


Fig. 10.30 3D Occupancy grids of the scenario defined as 3). Different perspectives of view.

This scenario has about 300 metres of length. It is the main corridor of the Faculty of Engineering of the University of Porto. It is a large corridor, with about 5 metres on its width and height. It is also constituted with circular columns. As it is the main corridor of FEUP, which actually has about 8 thousands of students, it is a really dynamic scenario, with an enormous number of students crossing it per day.

These three scenarios have enormous areas, but the *pre-localization* and *mapping* were applied in smaller areas of the buildings, resulting in smaller occupancy grids. Therefore, these smaller occupancy grids were merged aiming to obtain the bigger maps, i.e. occupancy grids, which are shown in this Chapter. This merging allowed the localization of the vehicle, during its normal operation, in these enormous maps.

The occupancy grid, in different perspectives of view, about the fourth scenario, is shown in figure Fig. 10.31. This corridor is about 60 metres long of length, 6 metres of width and height. It has lot of information in transversal and translational directions as is example of square columns.

As the mapping happened during the Portuguese National Festival of Robotics, "Robótica 2012", a lot of people were continuously crossing the environment that was intended to map.

Therefore, the *pre-localization* and *mapping* procedures were performed during the night, becoming the scenario complete controlled, without people or dynamic objects crossing it.

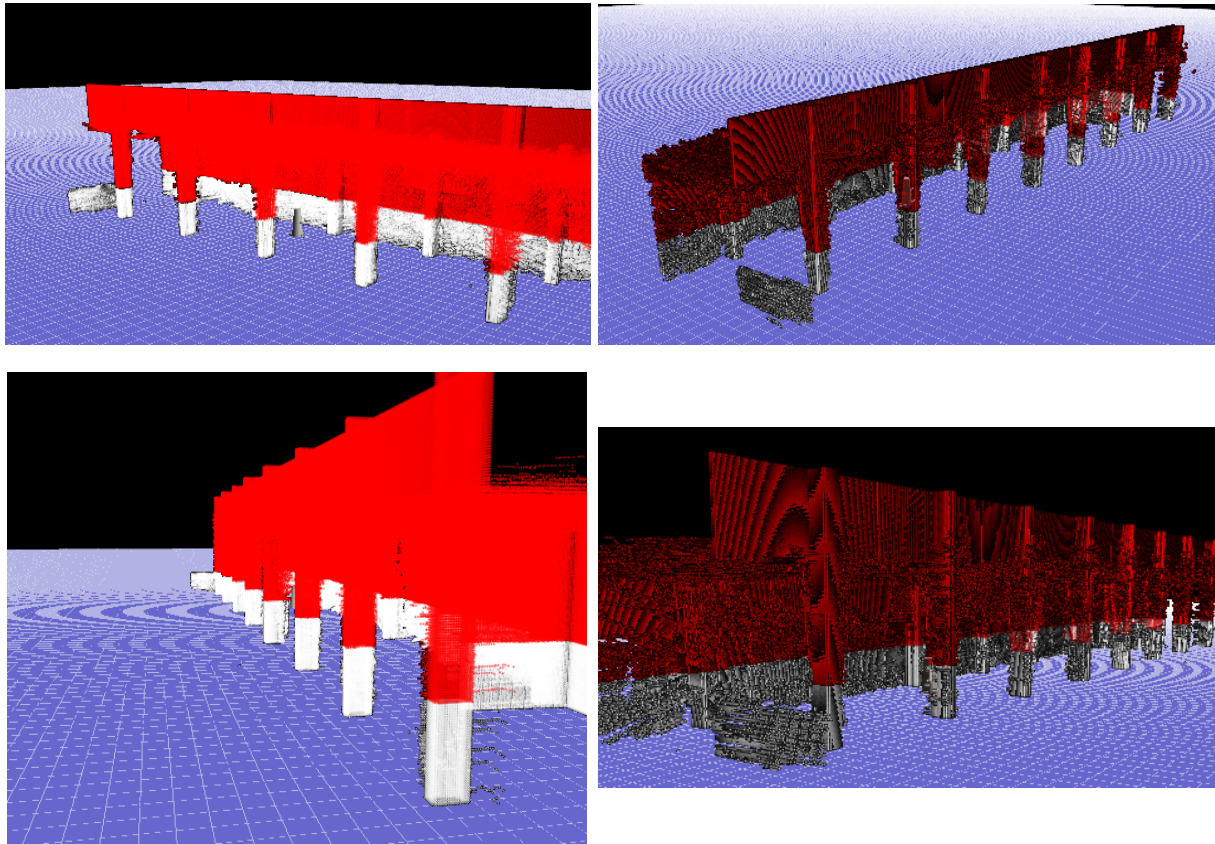
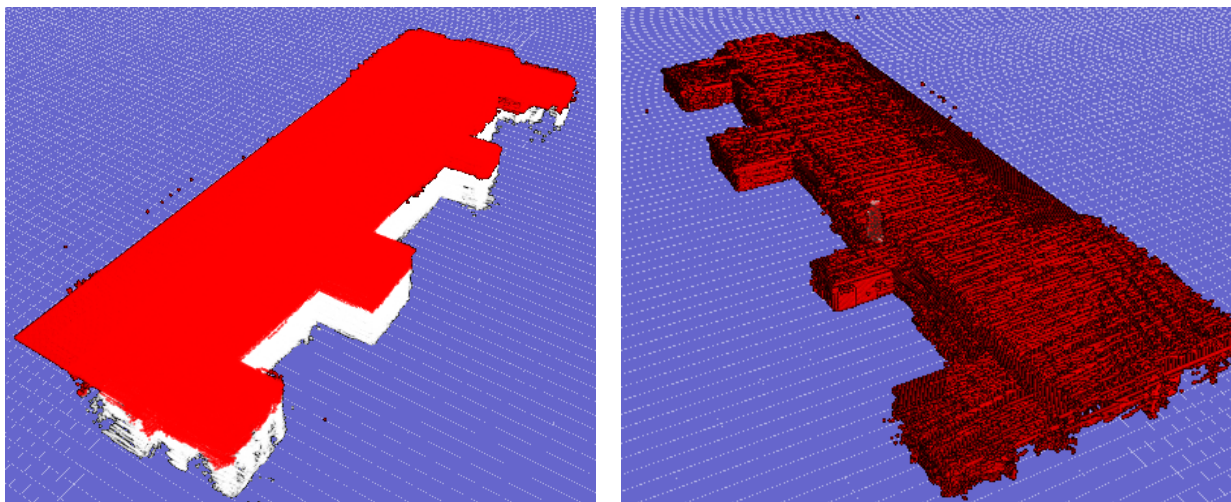


Fig. 10.31 3D Occupancy grids of the scenario defined as “pavilhão multiusos de guimarães”. Different perspectives of view.

The fifth scenario is represented by the 3D occupancy grids shown at the following figure, Fig. 10.32. This scenario is a small 25 metres long corridor with four columns. As it can be seen by presented 3D occupancy grids, this scenario have halls' entrances and circular columns, which really helpful for the vehicle location.





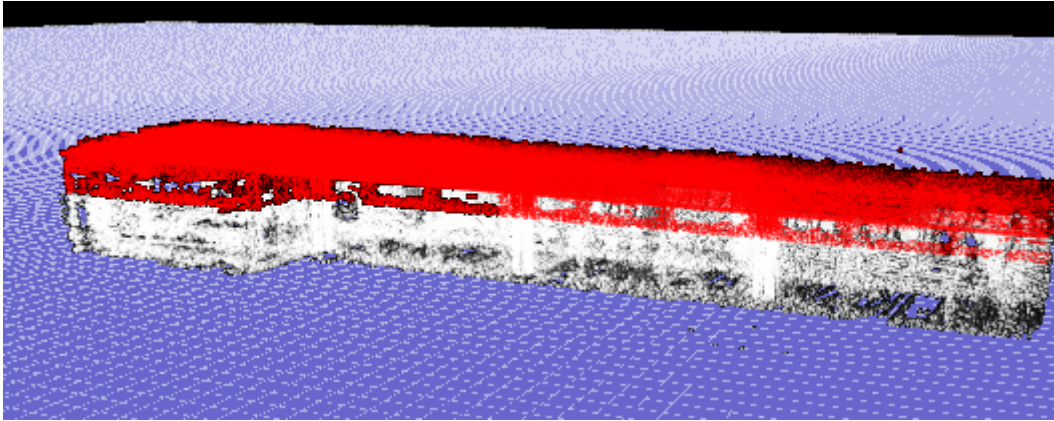


Fig. 10.32 3D Occupancy grids of the scenario defined as 5. Different perspectives of view.

The occupancy grid, represented in different perspectives of view, about the scenario defined above as 6), is shown in the Fig. 10.33. This is a small corridor with about 25 metres of length and circular columns, as can be seen in the following images. In this scenario, there are areas, where, due the small distance range of the tilting Laser Range Finder, the vehicle is not able to "see" helpful information for its localization. Therefore, in this scenario, the vehicle should follow a trajectory, where it is capable of acquire helpful data, to be used on its localization.

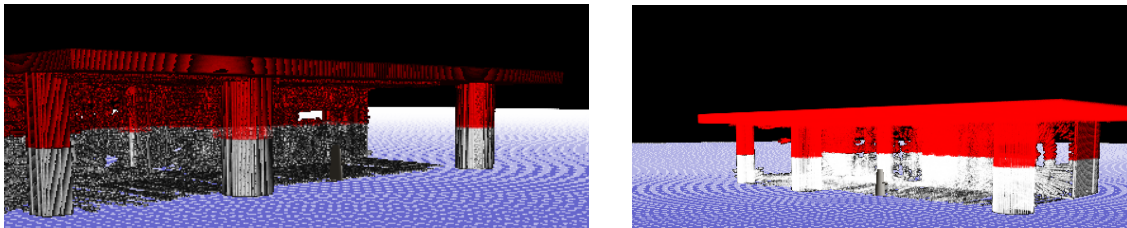
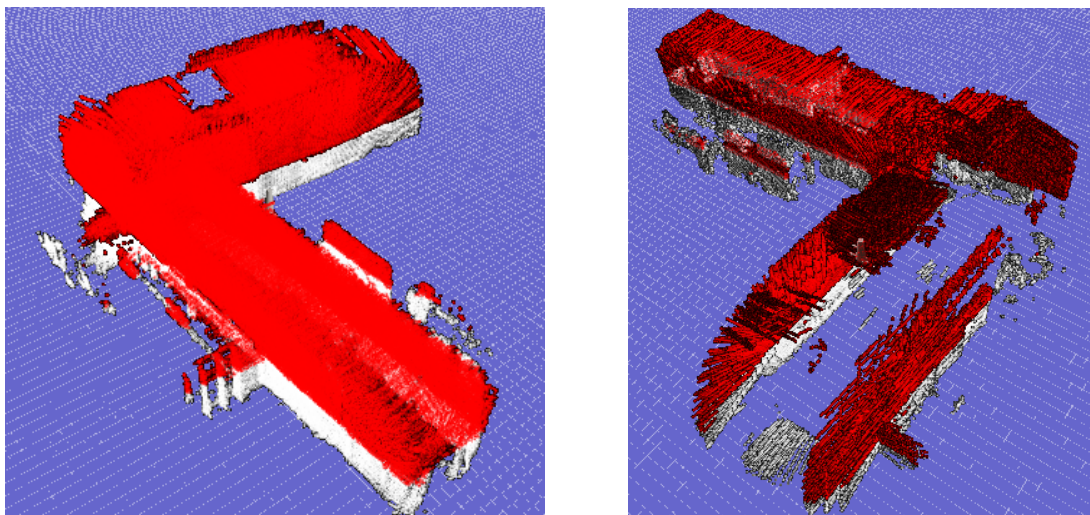


Fig. 10.33 Occupancy grids of the scenario defined as Exponor building during the "Fórum do Mar", scenario 6). Different perspectives of view.

The occupancy grid in the three-dimensional space, about the scenario defined above as scenario 7), the shopping gallery of the enterprise centre "Lionesa", in different perspectives of view, is shown in the following images, Fig. 10.34.



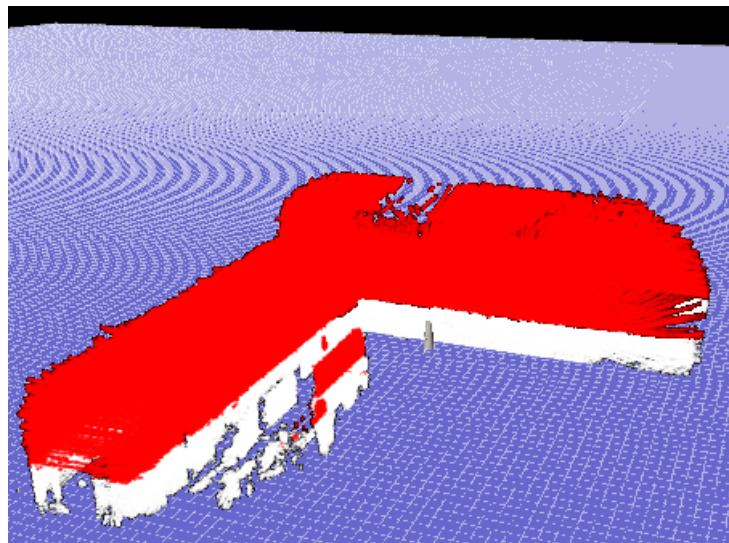


Fig. 10.34 3D Occupancy grids of the scenario defined as 7). Different perspectives of view.

This scenario was tested during a public demonstration. The environment is a gallery shopping, with people walking continuously. The mapped area has a square shape of 60 metres per 60 metres . The gallery has two corridors, forming the "L" word. The building ceiling has an height of about 5 metres and in certain zones has more, becoming impossible to be detected with the used Laser Range Finder, the *hokuyo URG-UG01*, which has a distance range of 5 metres .

The occupancy grid about the scenario defined above as 8), where this localization methodology was tested in a public demonstration, is shown at the following figure. This scenario has a square shape of 30 metres x 30 metres.

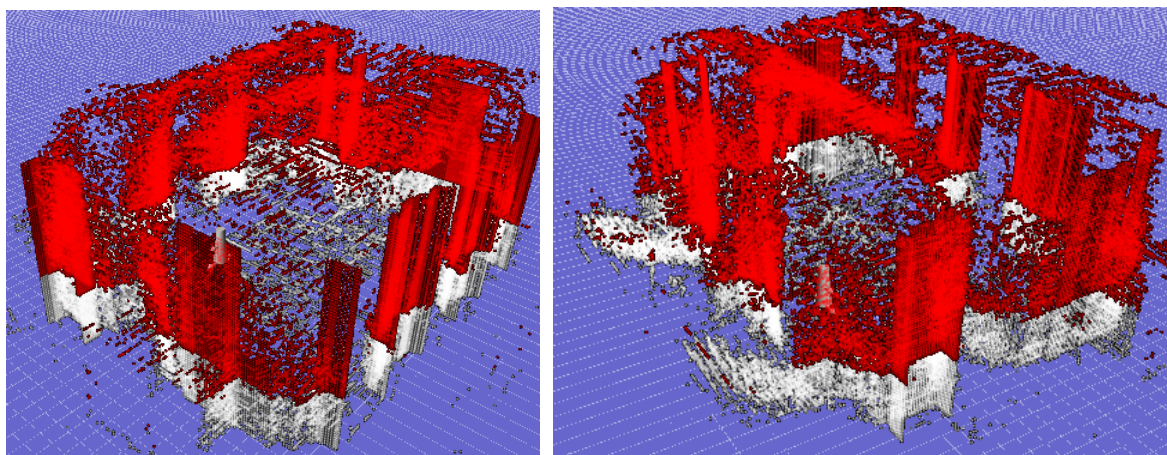


Fig. 10.35 3D Occupancy grids of the scenario defined as reitoria's lobby, scenario 8). Different perspectives of view.

In certain zones, the reitoria's lobby, has about six metres of height. In other zones, it is impossible the tilting Laser Range Finder to see the ceiling, because this building has about seven metres or more of height.

The walls of this building have lot of details and corners to be used in the localization, as it can be seen in the 3D occupancy grid presented below. Although being an environment

with some movement and people walking during the day, the *pre-localization* and *mapping* procedures were applied with log data acquired during an afternoon.

### 10.3.2. Distance and Gradient Matrices

After obtained the occupancy grid about the upper side of a building, as is example the three dimensional occupancy grids presented in the last sub-section, it is possible to apply the distance transform, as described in the sub-section 8.2, to compute the distance matrix of the mapped building,  $dmap$ .

Furthermore, after computed the distance transform, the Sobel filters, as described in the sub-section 8.2, are applied to obtain the gradient matrices, in both the directions  $x$  and  $y$ ,  $\nabla x_{3D}$  and  $\nabla y_{3D}$ , respectively.

These three matrices, the distance and gradients, are in the three-dimensional space and are stored in the memory, to be used during the *localization* algorithm, as described in Chapter 9. As already refereed these three matrices are the base core of the 3D Perfect Match algorithm, used during the vehicle normal operation. The use of pre-computed matrices becomes the localization algorithm really fast allowing to be used on-line.

The Fig. 10.36 shows a slice of the 3D distance matrix when the  $z$  coordinate is 1.8 metres,  $dmap(x, y, 1.8)$ . As can be seen in the figure, the darker zones are corresponding to cells whose distance to obstacles (as walls, corners, columns, ceiling, doors *et cetera*) is closer. The lighter zones, in the Fig. 10.36, are representative of cells which are more distant from obstacles. The Fig. 10.37 shows a slice of the 3D gradient matrix in the  $x$  direction, when the  $z$  coordinate is 1.8 metres,  $\nabla x_{3D}(x, y, 1.8)$ . On the contrary, the Fig. 10.38 shows a slice of the 3D gradient matrix in the  $y$  direction, when the  $z$  coordinate is 1.8 metres,  $\nabla y_{3D}(x, y, 1.8)$ . These matrices are about the occupancy grid shown in Fig. 10.25.



Fig. 10.36 Distance matrix slice in the height of 1.8 metres.  $dmap(x, y, 1.8)$ .

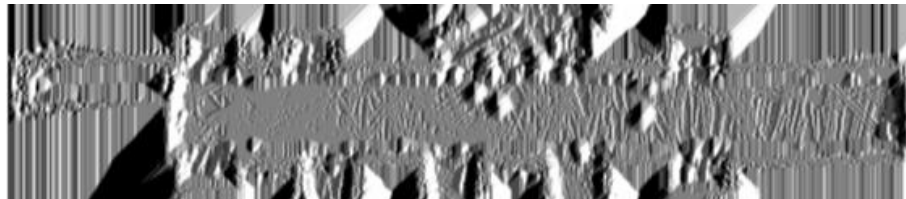


Fig. 10.37 Gradient matrix slice, in the  $x$  direction, to the height of 1.8 metres.  $\nabla x_{3D}(x, y, 1.8)$ .

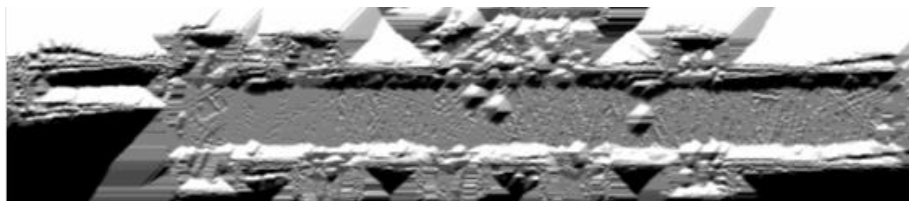


Fig. 10.38 Gradient matrix slice, in the  $y$  direction, to the height of 1.8 metres.  $\nabla y_{3D}(x, y, 1.8)$ .



With these matrices, shown in Fig. 10.37 and Fig. 10.38, it can be seen that the darker zones are representative of cells where the distance matrix decrease with the positive variation of  $x$  or  $y$ , respectively. On the contrary, the lighter zones are corresponding to cells where the distance matrix increase with the positive variation of  $x$  or  $y$ , respectively.

In this sub-section, the following figures are demonstrative of the distance and gradient matrices about different scenarios. Since, it is hard to represent the entire three-dimensional matrix of distances and gradients, only it is shown a slice, when the  $z$  coordinate is equal to 1.8 metres. The distance and gradients shown in this Chapter are equalized matrices, therefore, the lower value is represented in black, on the contrary, the higher value is represented in white.

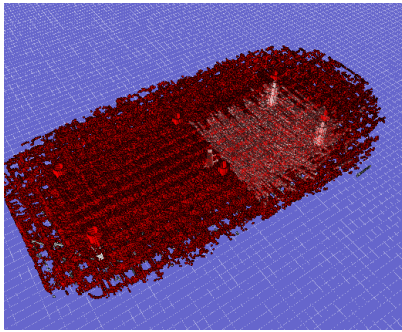


Fig. 10.39 Occupancy grid of part of the scenario 2)

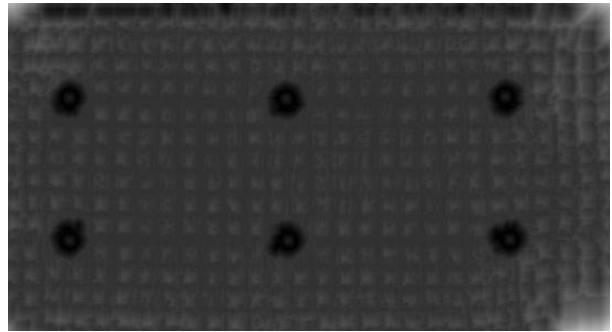


Fig. 10.40 Distance matrix slice in the height of 1.8 metres,  $dmap(x, y, 1.8)$ . It is about the map of Fig. 10.39.

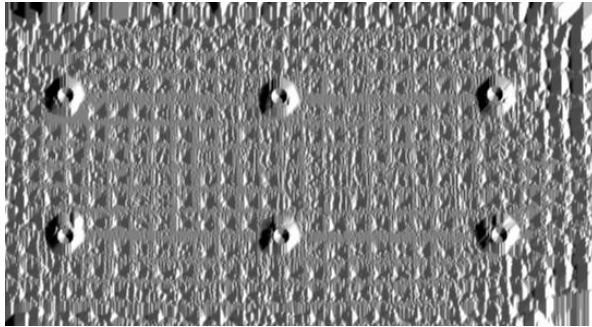


Fig. 10.41 Gradient matrix slice, in the  $x$  direction, in the height of 1.8 metres,  $\nabla x_{3D}(x, y, 1.8)$ . It is about the occupancy grid presented in Fig. 10.39.

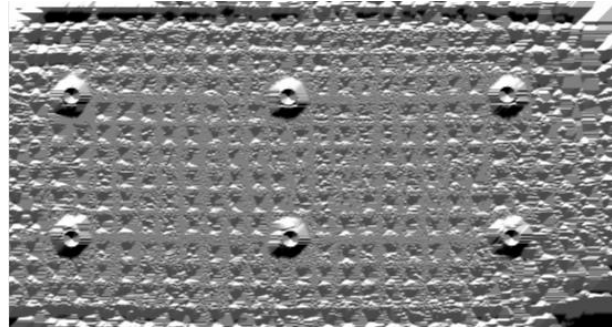


Fig. 10.42 Gradient matrix slice, in the  $y$  direction, to the height of 1.8 metres.  $\nabla y_{3D}(x, y, 1.8)$ , about the occupancy grid presented in Fig. 10.39.

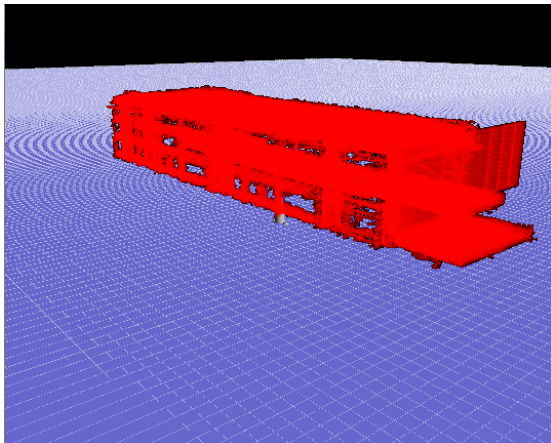


Fig. 10.43 Occupancy grid of part of the scenario 3).

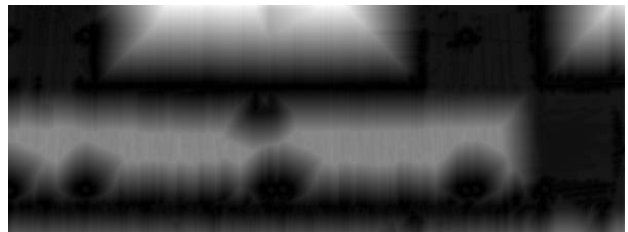


Fig. 10.44 Distance matrix slice for the height of 1.8 metres, about the occupancy grid presented in Fig. 10.43.



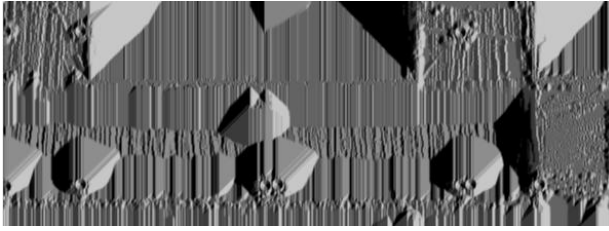


Fig. 10.45 Gradient matrix slice for the x direction, in the height of 1.8 metres. About the occupancy grid of Fig. 10.43.

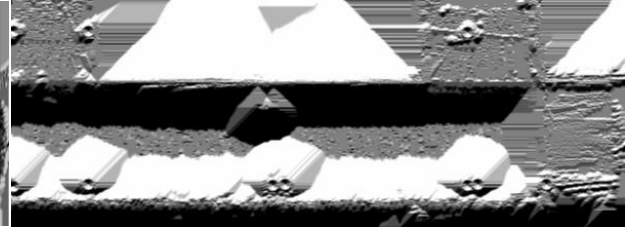


Fig. 10.46 Gradient matrix slice for the x direction, in the height of 1.8 metres. About the occupancy grid of Fig. 10.43.

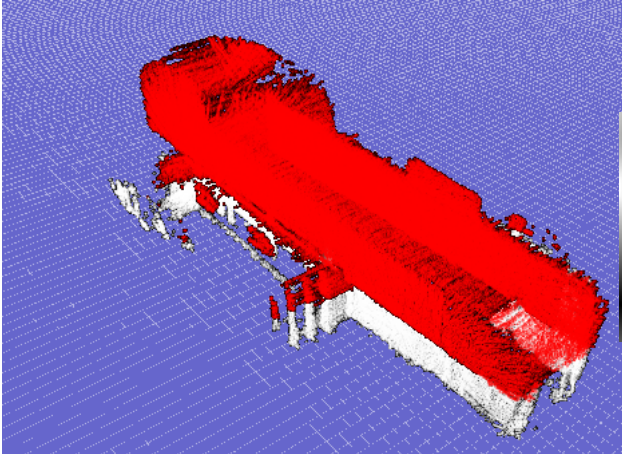


Fig. 10.47 Occupancy grid of part of the scenario 7)

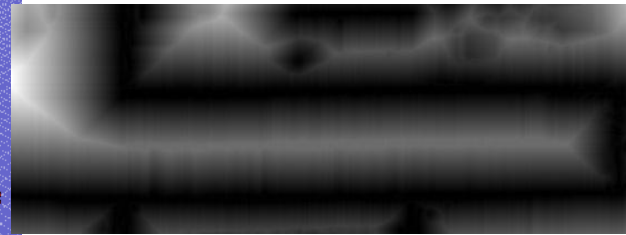


Fig. 10.48 Distance matrix slice in the height of 1.8 metres.  $d_{map}(x, y, 1.8)$ , about the occupancy grid of Fig. 10.47.

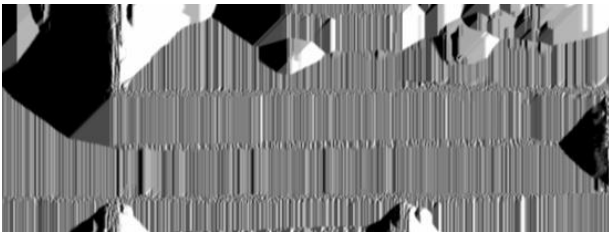


Fig. 10.49 Gradient matrix slice for the x direction, in the height of 1.8 metres. About the occupancy grid of Fig. 10.47.



Fig. 10.50 Gradient matrix slice for the x direction, in the height of 1.8 metres. About the occupancy grid of Fig. 10.47.

## 10.4. Localization Results

The 3D Matching algorithm takes an average execution time of 12 milliseconds to find the optimal solution, running in a Mini ITX, EPIA M10000G with a 1.0GHz processor, with 682 points of the tilting Laser Range Finder. This is very fast comparatively to the ICP, SLAM 2D or SLAM 3D algorithms. Therefore, it can be executed online, with a cycle period of 100 milliseconds imposed by the observation module sample rate. The execution time of the *localization* algorithm is detailed explained in the sub-section 10.4.4. Also a comparison with other works in the literature is done in this sub-section.

Other quality of the *localization* algorithm proposed here, with given proves in many exhaustive tests, are the no need of prepare the environment, with for example artificial

landmarks placed in the navigation scenario, using only odometry and the tilting LRF (*observation module*).

This algorithm of *localization* is only applied in structured indoor environments as the scenarios presented in the previous sub-section, 10.3.1. The localization algorithm execution time is independent of the size of the scenario. Therefore, the localization is able to be executed online in scenarios with any dimensions, as opposite to SLAM algorithms. To do that, it is necessary the three-dimensional distance and gradient matrices about the entire building.

It is also needed a pc with the necessary onboard memory to perform the loading of these matrices. A scenario with a square shape of  $25 \text{ metres}^2$  and an height of  $4 \text{ metres}$ , as is example the scenario defined as 2), Fig. 10.28, represented in a resolution of  $0.04 \text{ metres}$ , will require the following memory for the three matrices (distance and gradients):

$$\text{memory} = 3 \cdot \text{numBytes} \cdot \frac{25^2 \cdot (4 - \text{minH})}{0.04^3} \text{ (Bytes)} \quad (10.5)$$

in which  $\text{minH}$  is the height when the upper side of the building starts and is equal to  $1.8 \text{ metres}$ , since the points of  $P$  whose  $z$  coordinate is above  $1.8 \text{ metres}$  constitute the set of data points  $P_{3D}$ , with coordinates  $(x, y, z)$ , which are used during the matching localization procedure. As each component of the matrices is an float, the number of bytes ( $\text{numBytes}$ ) is equal to 4. Therefore, the previous equation can be re-written as follows:

$$\begin{aligned} \text{memory} &= 3 \cdot 4 \cdot \frac{25^2 \cdot (4 - 1.8)}{0.04^3} \text{ (Bytes)} \Leftrightarrow \\ \text{memory} &= 257.81 \text{ MegaBytes} \end{aligned} \quad (10.6)$$

In some situations, aiming a better usage of the pc's memory, a map can be divided in smaller maps, whose relation between them is well known. For example, consider the map with an configuration as the one defined as the scenario 7), shown in Fig. 10.34, whose shape is like an "L". This map is inside a square of  $60 \text{ meters}^2$  of area. But, the navigation area is much more smaller than  $60 \text{ meters}^2$  (it is only inside the "L"). Therefore, inside this square shape with an area equal to  $60 \text{ meters}^2$ , there is a dead zone, occupying unnecessary memory, that will not be used for the vehicle localization algorithm, since vehicle cannot navigate in that zone of the map. Therefore, it is preferable to divide such map in two, one for each leg of the "L" and therefore, economize on the memory needed to navigate in a scenario like that.

The algorithm of localization was exhaustively tested in the eight scenarios described above. The robot is able to navigate in each of this scenarios, in a complete autonomous way, between waypoints. The robot RobVigil has a GUI interface, called "rondas", that allows an user to pre-define surveillance routines, to be executed by the robot in an autonomous way.

The GUI interface "rondas" is shown in Fig. 10.51. In this case the loaded map is the corresponding to scenario described at the beginning of this Chapter as 1). The red squares represent the waypoints that the vehicle is intended to cross. Between waypoints, the trajectory intended is shown with a green line.

In the eight scenarios defined above, were made public demonstrations and exhaustive tests. These demonstrations and tests were made with the vehicle performing completely autonomous surveillance routines, pre-defined with waypoints marked by an user.

Therefore, to each of these eight scenarios, the GUI interface "rondas" uses the architectural plan of the building. To perform autonomous surveillance routines, to each of these scenarios, it was necessary to compute a relation between the architectural plan and the map used by the robot to perform its localization, based on the *three-dimensional map-based* approach. For each of the eight scenarios a relation was established based only in the scale and the translational and rotational offsets between the maps. This fact, proves the correctness of the maps obtained during the *pre-localization* and *mapping* phase.

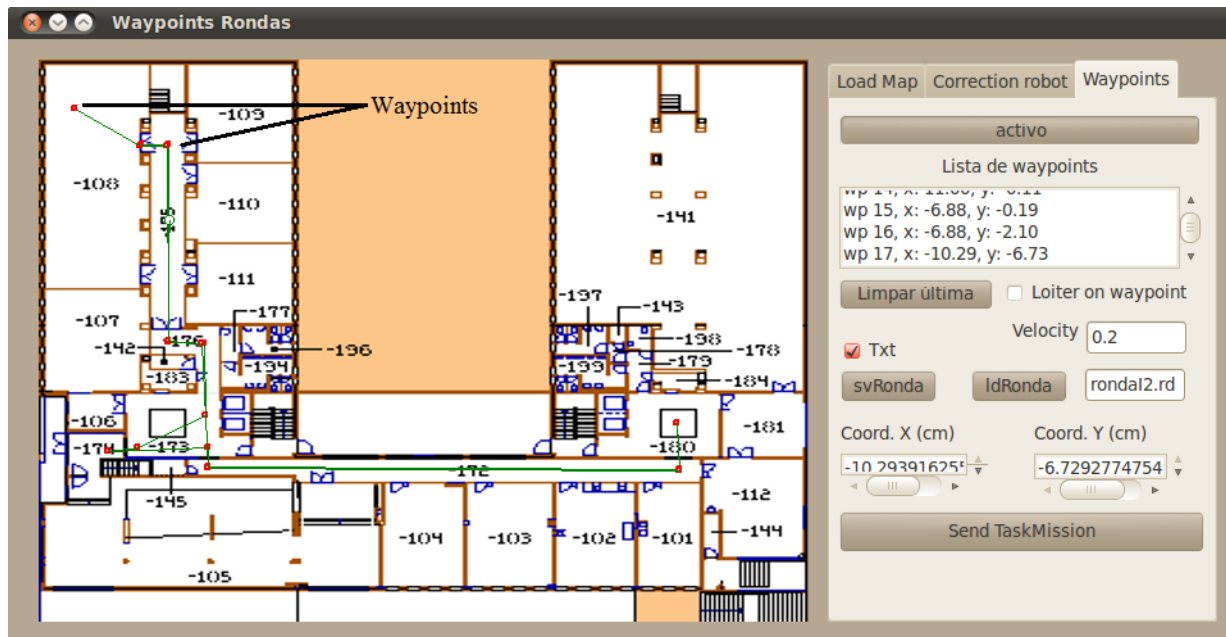


Fig. 10.51 GUI interface "rondas". It shows the architectural plan. The marked waypoints are shown in red. The green line shows the trajectory that it is intended to follow.

The following table, Table 10.2 summarizes the set of public demonstrations and tests performed in the eight scenarios. In these tests, the vehicle performed during consecutive hours surveillance routines in a completely autonomous way. It should be highlighted the success that the robot had, during these public demonstrations. News about the RobVigil appeared on the Portuguese television channels: RTP1, SIC and TVI. Also in the "Exame Informática" Magazine and in the "Jornal de Notícias" newspaper, interviews about the robot and public demonstrations performed, were shown.

Between the public demonstrations presented in the following table, it should be highlighted the demonstrations at the Engineering weekend, during the festival of "Robótica 2012", during the "Fórum do Mar" fair at the Exponor, the demonstration in the UTM unit day at the INESC Porto building; and finally the demonstration during the Europeans Researchers night, at the "Reitoria" of the University of Porto. In these demonstrations the vehicle performed during consecutive hours, surveillance routines, even in the presence of lot of curious people around it.

Scenario	Demo/Test Duration	Event
1	One morning	"Exame Informática" Magazine
	One morning	News on the RTP1 .
	One Afternoon	Engineering weekend.
	Months	Exhaustive debugging tests
2	Two mornings	Exhaustive debugging tests
3	Two mornings	Exhaustive debugging tests
4	Four days	"Robótica 2012"
	One morning	FreeBots competition "Robótica 2012"
5	One Afternoon	UTM unit day at the INESC Porto building
6	Three days	"Fórum do Mar" at Exponor
7	One Afternoon	Shopping gallery of the enterprise centre "Lionesa"
8	One night	"Reitoria" of the University of Porto, Europeans Researchers night.

Table 10.2 Demonstrations and exhaustive tests performed by the robot during events and debugging tests.

The following figures show the vehicle trajectory (position) during experimental tests performed in these eight scenarios. During this experimental tests the speed of the robot was 0.4 m/s.

In figures, Fig. 10.52 to Fig. 10.59, at yellow is shown the vehicle localization obtained using the 3D Matching *localization* algorithm, which pinpoint the vehicle position with success. In green is shown the vehicle position computed only using the odometry. As can be seen in the entire set of tests, the odometry trajectory ever diverges from the correct obtained with the 3D Matching algorithm.

This trajectories were made using remote control (joystick) aiming to test the *localization* robustness. The sudden curves and loops, which are possible to see in the previous images,

were made purposely to test the robustness of the 3D Matching localization algorithm. As it is possible to see also, on the images, the vehicle operated correctly in all the trajectories.

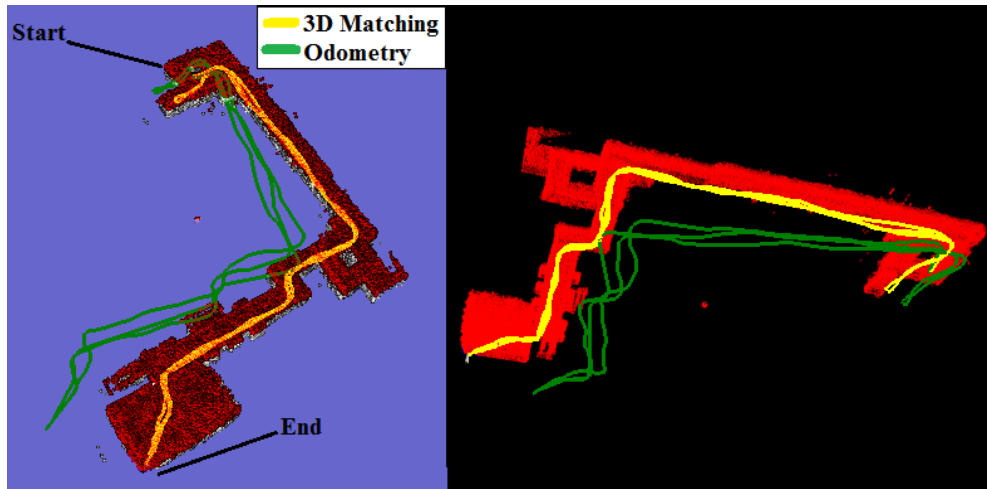


Fig. 10.52 The trajectory of the vehicle, Start-End-Start-End, in the scenario 1). In yellow the correct position (3D Matching). In green the odometry position. In the left image the 3D map is seen overhead. In the right image it is seen underneath.

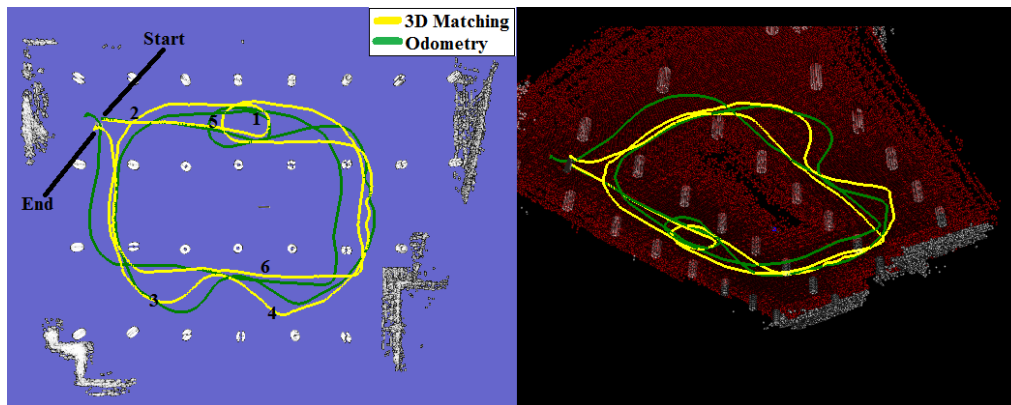


Fig. 10.53 The trajectory of the vehicle, Start-1-2-3-4-5-6-End, in the scenario 2). In yellow the 3D Matching's position. In green the odometry. In left, the image is seen overhead. In the right image it is seen underneath.

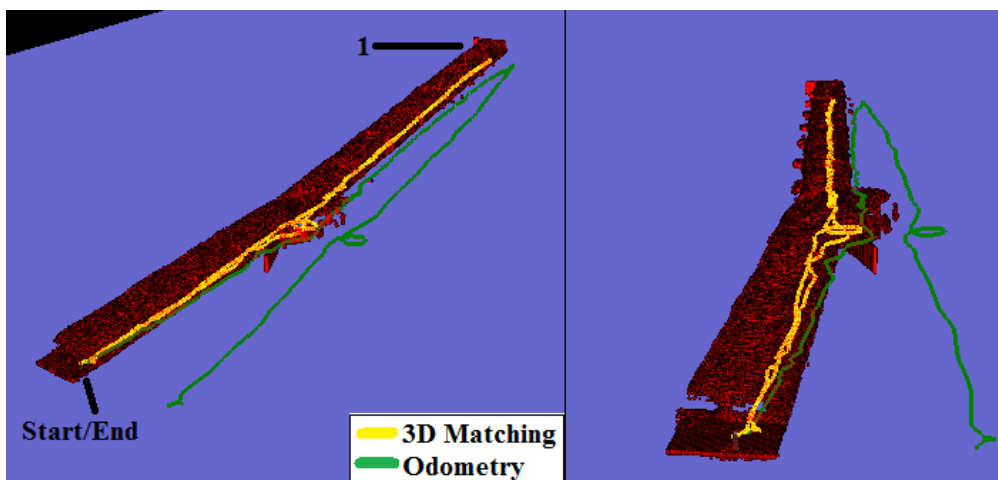


Fig. 10.54 The trajectory of the vehicle, Start-1-End with loops at the middle, in the scenario 3). In yellow the 3D Matching's position. In green the odometry. Different perspectives of the trajectory.

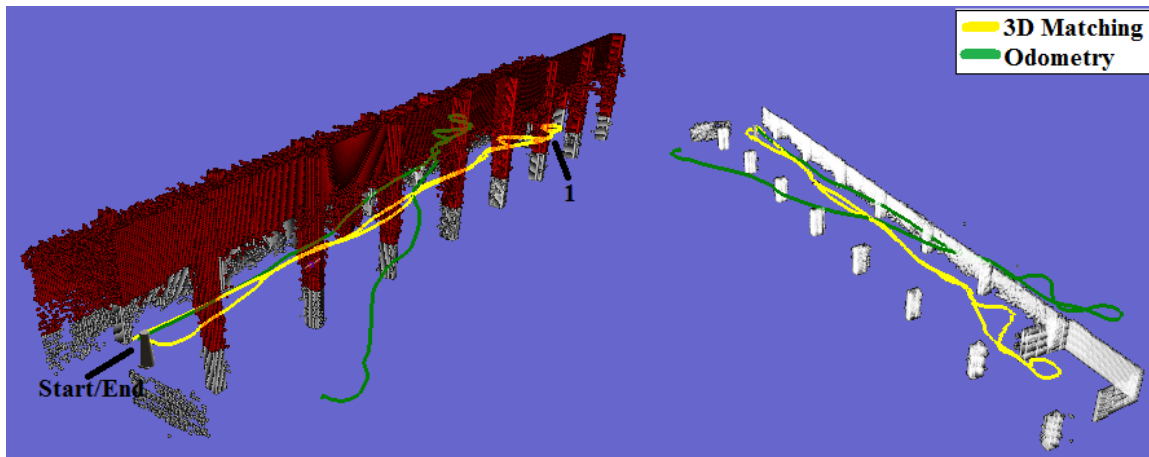


Fig. 10.55 The trajectory of the vehicle, Start-1-End, in the scenario 4). In yellow the 3D Matching's position. In green the odometry. Left: the entire occupancy grid is shown. Right, only the bottom part (white, below 1.8 metres) is shown. The red part is used for vehicle localization (3D Matching).

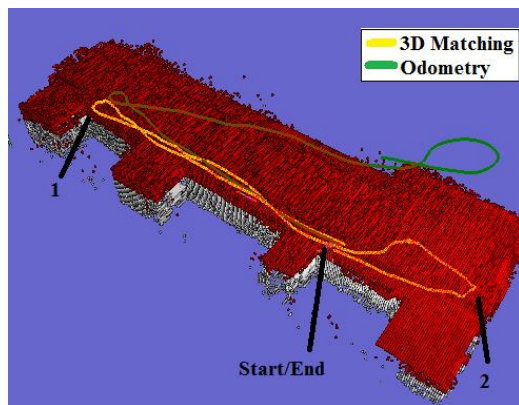


Fig. 10.56 The vehicles position during the trajectory , Start-1-2-End, in the scenario 5) is shown in yellow, through the localization obtained using the 3D Matching. In green the odometry trajectory.

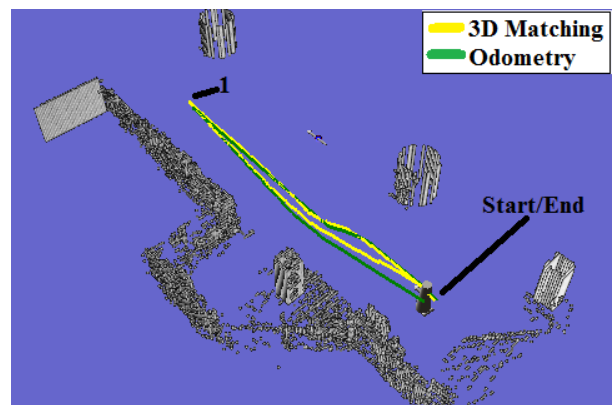


Fig. 10.57 The trajectory of the vehicle, Start-1-End, in the scenario 6). In yellow the 3D Matching's position. In green the odometry. Only the bottom part of the occupancy grid (below 1.8 metres) is shown with white points.

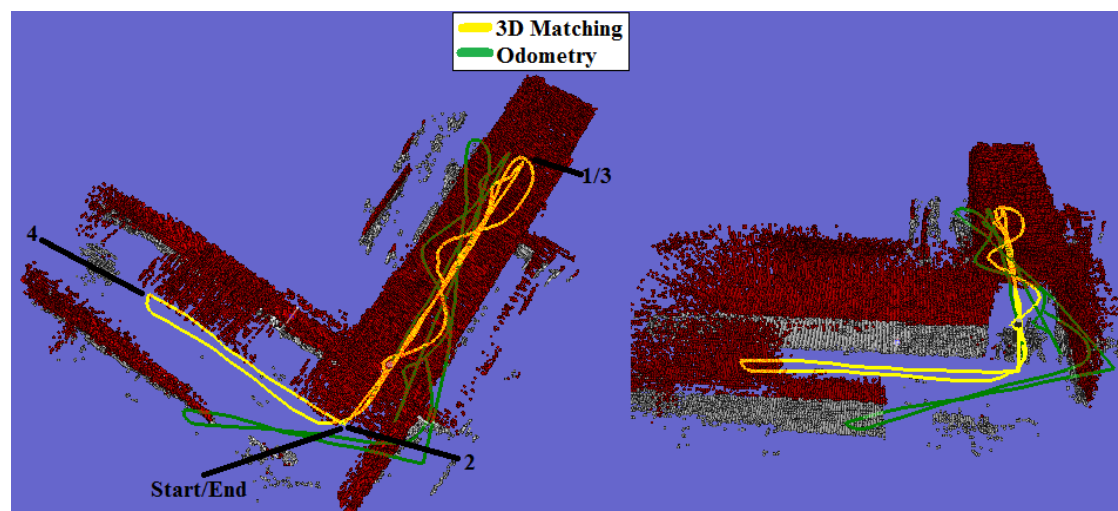


Fig. 10.58 The vehicle's trajectory, Start-1-2-3-4-End, in the scenario 7). In yellow the 3D Matching's position. In green the odometry. Different perspectives of the trajectory.



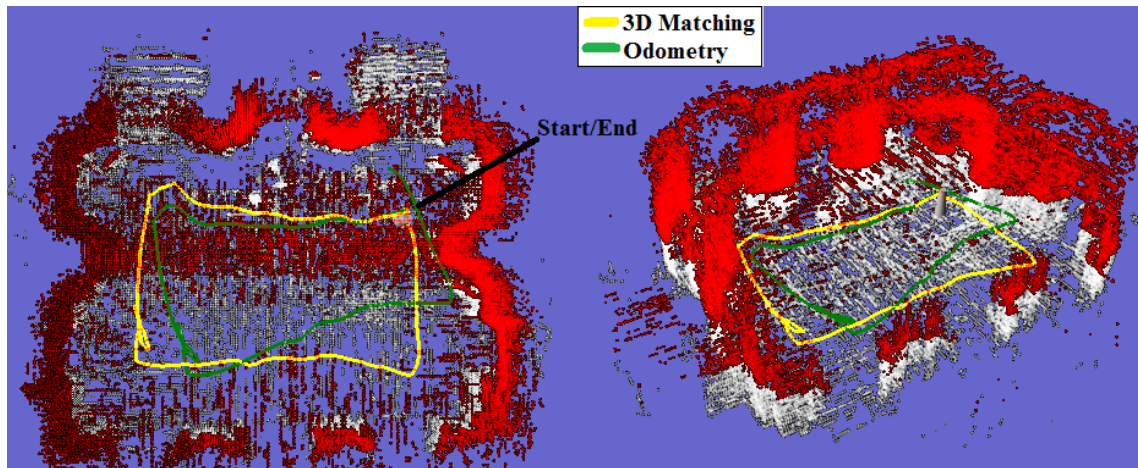


Fig. 10.59 The vehicle's trajectory from the label Start to the label End, in the last scenario 8). In yellow the 3D Matching's position. In green the odometry. Different perspectives of the trajectory.

#### 10.4.1. Reference Localization System Sick Nav350

The Sick Nav350 is a laser positioning system developed and commercialized by the Sick Sensor Intelligence. This sensor is applied in autonomous guided vehicles (AGVs) with reflectors placed in the navigation area. Some specific scenarios referred by the sensor's developers are: truck loading, shuttle systems and autonomous systems using AGVs. The Nav350 sensor and a reflector are shown in figures: Fig. 2.8 and Fig. 2.9.

The Nav350 allows RS-232 or Ethernet communication, it weighs approximated 2.4 Kg and is mechanical robust. The sensor is able to detect high reflective reflectors, planar or cylindrical. The reflectors should be covered with a high reflective pellicle provided by the sensor's developers.

Some of features of the Sick Nav350, which are important to highlight are: 1) it is able to measure at a maximum of 40 reflectors; 2) the angle sensor's field of view is 360°; 3) the sensor is able to detect reflectors between 0.5 metres to 70 metres; 4) it rotates with a frequency of 8Hz; 5) it uses the reflectors to output its self location; 6) a location position is available with a sample rate of about 125 milliseconds, with an accuracy of 4 millimetres to 25 millimetres, depending of the distance and relative position between reflectors, that should be carefully configured; 7) the sensor also provides, the contour of the surrounding scenario, in a range distance of 70 metres and with an angular accuracy of 0.1 degrees; finally, 8) it is an expensive sensor.

The sensor has a protocol of communication, which allows sending to it, odometry data, acquired by the vehicle's odometers and; with an frequency of 8Hz obtain its self location in the absolute referential; the reflectors positions, in Cartesian or polar coordinates, in relation to the sensor referential; and finally the surrounding environment contour (distance and angle of each beam), also in relation to the sensor referential.

Aiming to compare the accuracy of the localization algorithm based on the 3D Matching presented in this PhD document, described in Chapter 9, with the Nav350, experimental tests were performed.

Therefore, in the three first scenarios, described in the beginning of this Chapter, the *localization* result in the *three-dimensional map-based* approach, was compared with the

localization provided by the Sick Nav350. To perform this comparison, was developed a GUI with a communication interface, using Ethernet and the Nav350 protocol. Also six cylindrical reflectors were constructed and placed along the scenarios, allowing the Nav350's self location.

The reflectors were constructed considering the following considerations, referred by the sensor's developers: 1) cylindrical reflectors need to be seen in any angle (should be cylindrical); 2) the reflectors should be covered with a reflective tape, provided together with the sensor, which has an high reflectance; 3) the cylindrical reflectors should have a minimum diameter, equal to 80 millimetres; 4) the size of the reflectors should be at least 500 millimetres, when is intended to be seen at a minimum distance of 30 metres.

Also several considerations should be taken in account during the reflectors distribution along the navigation area. To a precise location the Nav350 in any position should see 4 to 5 reflectors, but 3 are sufficient to compute its self location. The minimum distance to avoid confusion or bad association between reflectors is 300 millimetres. The reflectors should not be in positions where there are overlapping, between them. Finally, the reflectors should not be placed symmetrically in the scenario.

In the first scenario, defined above as 1), there are a corridor whose floor is a grid with known dimensions. Therefore, the ground truth is only available in this corridor; and for that reason it was chosen to carry out a characterization of the Nav350 self localization algorithm. Therefore, if the Nav350 true position is well known, i.e. the ground truth  $x_{GND}$ ,  $y_{GND}$  and  $\theta_{GND}$  can be calculated using the corridor floor grid, the accuracy of the Sick Nav350 can be characterized.

The following figure, Fig. 10.60, shows one reflector and the Sick Nav350, placed on the robot RobVigil. In the Fig. 10.61, it is shown the corridor referred in the previous paragraph, where the red boxes highlight the presence of reflectors.

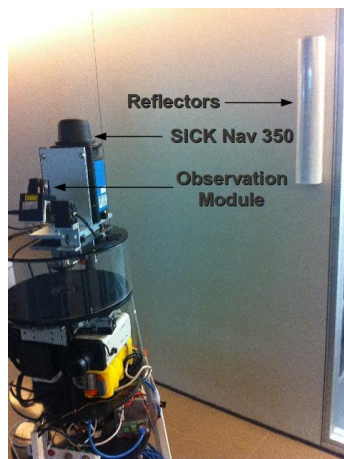


Fig. 10.60. Vehicle, observation module, Sick Nav350 and reflector.



Fig. 10.61. First scenario where tests of accuracy were performed.

In the next sub-section, it will be done a characterization of the accuracy of the *localization* algorithm proposed here, in relation to the Nav350. This characterization together with the accuracy characterization of the Sick Nav350 in relation to the GND, allows



achieving the final objective of characterize the accuracy of the localization algorithm proposed here, in relation to the GND.

The error distance  $\varepsilon_d$  and the angle error  $\varepsilon_{\theta_v}$ , between the reference position (*GND*) and the estimated position of the vehicle are given by:

$$\varepsilon_d = \sqrt{\varepsilon_{x_v}^2 + \varepsilon_{y_v}^2}, \quad \varepsilon_{x_v} = x_{GND} - x_v, \quad \varepsilon_{y_v} = y_{GND} - y_v \quad (10.7)$$

$$\varepsilon_{\theta_v} = \theta_{GND} - \theta_v \quad (10.8)$$

in which the variables  $x_{GND}, y_{GND}$  and  $\theta_{GND}$  are the ground truth (reference positions), obtained and  $x_v, y_v$  and  $\theta_v$  are the vehicle estimated variables.

The Table 10.3 represents the average and standard deviation of the errors  $\varepsilon_{x_v}$ ,  $\varepsilon_{y_v}$ ,  $\varepsilon_{\theta_v}$  and  $\varepsilon_d$ , between the Nav350 self localization algorithm and the ground truth.

$\varepsilon_{GND}^{Nav350}$	Localization			
	$\varepsilon_{x_v} (metres)$	$\varepsilon_{y_v} (metres)$	$\varepsilon_{\theta_v} (radians)$	$\varepsilon_d (metres)$
$\hat{\varepsilon} = 1/N \sum \varepsilon$	-0.00006	0.0026	0.003	0.011
$\sigma_{\varepsilon} = \sqrt{1/N \sum (\varepsilon - \hat{\varepsilon})^2}$	0.006	0.015	0.0037	0.017
$\hat{\varepsilon} + 2 \cdot [-\sigma_{\varepsilon}, \sigma_{\varepsilon}]$	[-0.013, 0.013]	[-0.028, 0.033]	[-0.0044, 0.0104]	[0, 0.034]

Table 10.3 Results of the average and standard deviation of  $\varepsilon_d$  and  $\varepsilon_{\theta}$ . Nav350 localization system referenced to ground truth data.

Considering Gaussian, the error between the Nav350 localization and the ground truth, with the analysis of the Table 10.3, can be said that the angle error  $\varepsilon_{\theta_v}$  is in the interval  $[-0.0044, 0.0104]$  radians, with a certainty of 95.4%. The distance error is in the 95.4% of the cases in the interval equal to  $[0, 0.034]$  metres.

### 10.4.2. 3D Matching Accuracy Characterization

After characterized the error of the Nav350 localization system it is possible characterize the error of the 3D matching algorithm in the three scenarios already described. In the worst case, the error of the 3D matching algorithm in relation to the GND can be computed with the following equation:

$$\varepsilon_{GND}^{3DMatching} \leq \varepsilon_{Nav350}^{3DMatching} + \varepsilon_{GND}^{Nav350} \quad (10.9)$$

in which  $\varepsilon_{Nav350}^{3DMatching}$  is the error between the *localization* algorithm, proposed in this PhD work, and the Nav350 localization system. The error  $\varepsilon_{GND}^{Nav350}$  is representative of the the Nav350 positioning system with reference to the ground truth, already presented in the Table 10.3. The error  $\varepsilon_{GND}^{3DMatching}$ , is the error between the 3D Matching algorithm of localization and the ground truth.

Consider the error between the Nav350 and the ground truth as a Gaussian distribution. Consider the error between the 3D matching localization and the Nav350 positioning system also Gaussian. In that way, using the equation (10.9), the resultant error between the 3D

matching localization algorithm and the ground truth, is also a Gaussian distribution, with an estimative equal to the following equation:

$$\hat{\epsilon}_{GND}^{3DMatching} \leq \hat{\epsilon}_{Nav350}^{3DMatching} + \hat{\epsilon}_{GND}^{Nav350} \quad (10.10)$$

and standard deviation given by the expression:

$$\sigma_{GND}^{3DMatching} \leq \sqrt{(\sigma_{Nav350}^{3DMatching})^2 + (\sigma_{GND}^{Nav350})^2} \quad (10.11)$$

Due the absence of more reflectors (there are only six available reflectors in the laboratory), the accuracy tests were made only in a part of each of these three scenarios.

Each scenario, as already referred, has big dimensions (the first and the second scenarios have a square shape with size 60 x 60 metres , while the third scenario has 300 metres of length). To perform the accuracy tests in the entire dimension of each scenario, there are necessary a lot of reflectors. Therefore, as only six reflectors are available, the accuracy test was made in smaller parts of each scenario.

The Fig. 10.62 shows the results in the first scenario, in which the green circles are representative of the reflectors used by the industrial location sensor Nav350. The black line is the Nav350 trajectory, while the blue line is the trajectory estimated by the localization algorithm described and proposed in this thesis.

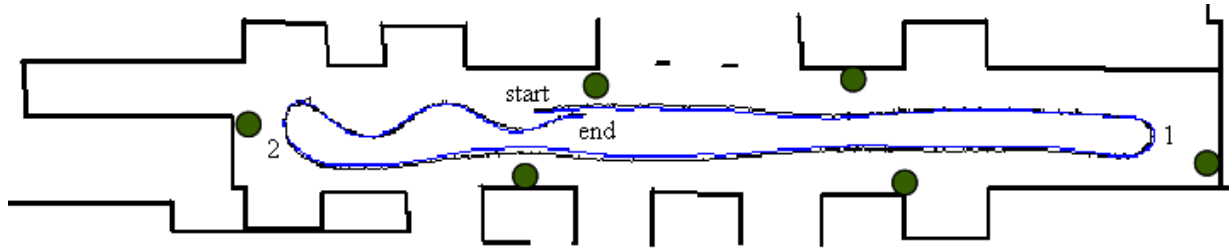


Fig. 10.62. Estimated location with reference with the Sick Nav350 position. The vehicle position represented at black is obtained with the Nav350. The blue vehicle trajectory is the estimated location. The green six circles are the artificial landmarks, i.e. reflectors placed by the scenario. The corridor has 25 metres long and 6 metres wide.

As can be seen, in Fig. 10.62, the estimation of the position (blue line) is close to the reference, Nav350 self location (black line).

The followed trajectory begins in the *start*, goes to the point *1*, *2* and ends in the *end*, as shown in Fig. 10.62.

The error between the localization algorithm proposed in this PhD thesis and the Nav350 positioning system is considered Gaussian. Therefore, with the analysis of the Table 10.4 it can be conclude that with an certainty of 95.4% ( $\hat{\epsilon} \pm 2 \cdot \sigma_{\epsilon}$ ), the angle ( $\epsilon_{\theta_v}$ ) and distance ( $\epsilon_d$ ) errors are into the intervals  $[-0.046, 0.059]$  radians and  $[0, 0.3]$  metres , respectively.

With the analyses of the Table 10.3 and Table 10.4 the new Table 10.5 can be computed. The Table 10.5 represent the error between the 3D matching localization system and the ground truth. The average of the error in this table is given by the sum of the average errors of Table 10.3 and Table 10.4, as shown in equation (10.10). The resultant standard deviation is computed using the equation (10.11).

$\varepsilon_{Nav350}^{3DMatching}$	Localization			
	$\varepsilon_{x_v} (metres)$	$\varepsilon_{y_v} (metres)$	$\varepsilon_{\theta_v} (radians)$	$\varepsilon_d (metres)$
$\hat{\varepsilon} = 1/N \sum \varepsilon$	-0.0284	-0.0118	0.0062	0.0957
$\sigma_{\varepsilon} = \sqrt{1/N \sum (\varepsilon - \hat{\varepsilon})^2}$	0.088	0.055	0.026	0.10
$\hat{\varepsilon} + 2 \cdot [-\sigma_{\varepsilon}, \sigma_{\varepsilon}]$	$[-0.20, 0.15]$	$[-0.12, 0.10]$	$[-0.046, 0.059]$	$[0, 0.3]$

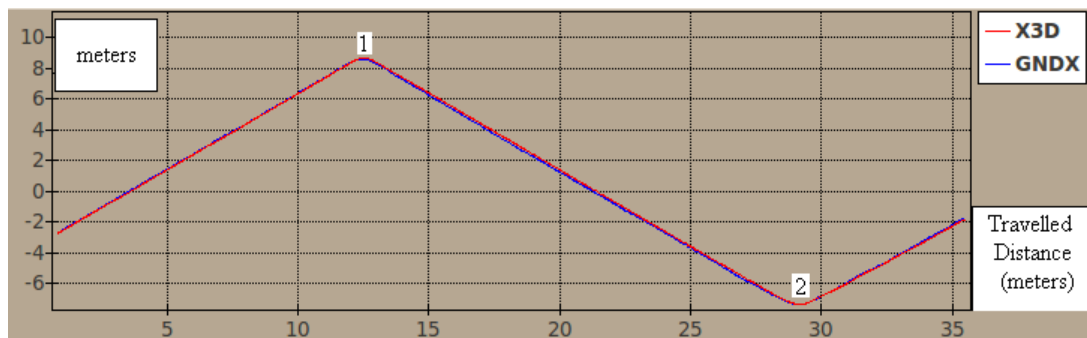
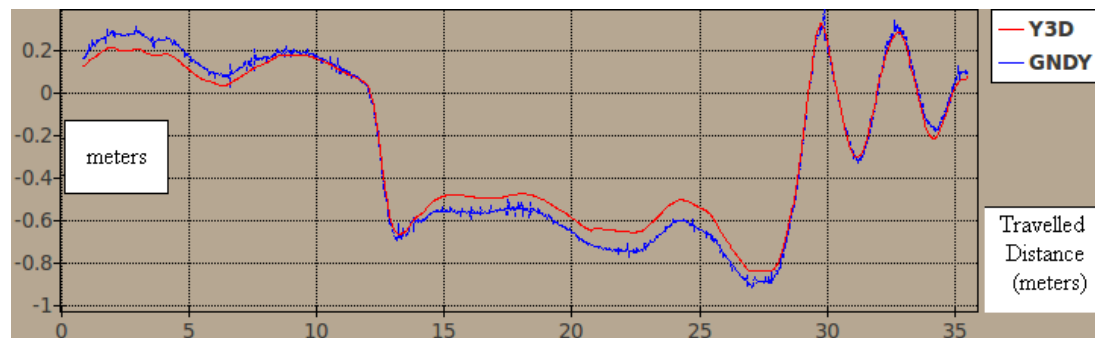
Table 10.4 Results of the average and standard deviation of  $\varepsilon_d$  and  $\varepsilon_{\theta}$ . Comparison between the 3D matching the Nav350.

Thus, the Table 10.5 helps to realize that in the 95.5% (two times the standard deviation) of the cases, the 3D matching localization errors,  $\varepsilon_{\theta_v}$  and  $\varepsilon_d$ , are inside the intervals,  $[-0.043, 0.062]$  radians and  $[0, 0.31]$  metres , respectively.

$\varepsilon_{GND}^{3DMatching}$	Localization			
	$\varepsilon_{x_v} (metres)$	$\varepsilon_{y_v} (metres)$	$\varepsilon_{\theta_v} (radians)$	$\varepsilon_d (metres)$
$\hat{\varepsilon} = 1/N \sum \varepsilon$	-0.028	-0.009	0.009	0.107
$\sigma_{\varepsilon} = \sqrt{1/N \sum (\varepsilon - \hat{\varepsilon})^2}$	0.088	0.057	0.026	0.101
$\hat{\varepsilon} + 2 \cdot [-\sigma_{\varepsilon}, \sigma_{\varepsilon}]$	$[-0.20, 0.15]$	$[-0.12, 0.10]$	$[-0.043, 0.062]$	$[0, 0.31]$

Table 10.5 Results of the average and standard deviation of  $\varepsilon_d$  and  $\varepsilon_{\theta}$ . Comparison between the 3D matching and GND.

As can be seen trough Fig. 10.63 to Fig. 10.66, the x position error is ever lower than 20 centimetres. The y position error is ever lower than 15 centimetres, while the  $\theta$  error is ever time lower than 0.1 radians and in almost the times lower than 0.05 radians.

Fig. 10.63  $x$  variable in function to the travelled distance. The vehicle estimated position  $x_v$  is represented in red, while in blue is shown the Nav350's  $x$  position.Fig. 10.64  $y$  variable in function to the travelled distance. The vehicle estimated position  $y_v$  is represented in red, while in blue is shown the Nav350's  $y$  position.

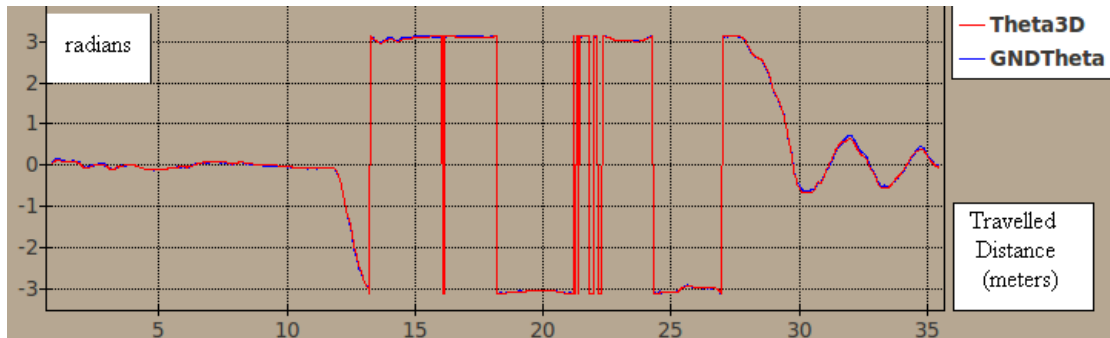


Fig. 10.65  $\theta$  variable in function to the travelled distance. The vehicle estimated position  $\theta_v$  is represented in red, while in blue is shown the Nav350's  $\theta$  position.

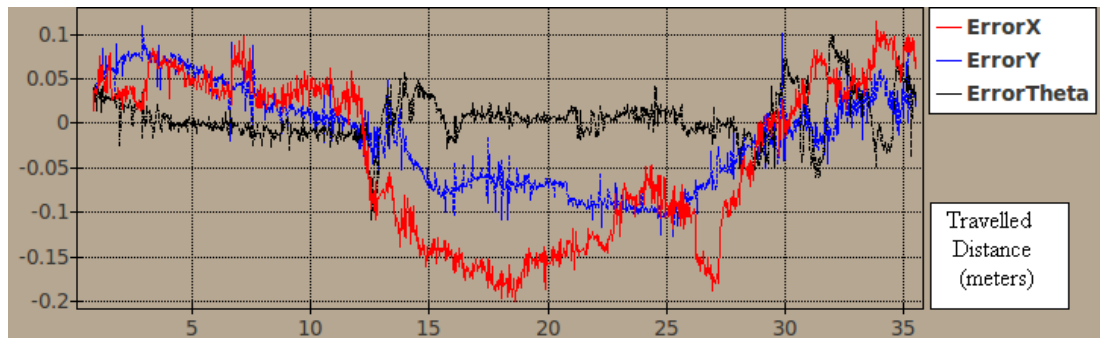


Fig. 10.66 Error curves between the 3D matching localization and the Nav350 positioning system estimation.

The Fig. 10.67 shows the comparison results in the second scenario defined at the beginning of this Chapter. As can be seen in Fig. 10.67 the localization's position estimation (blue line) is close to the reference, Nav350's self location (black line). The trajectory begins in the *start* point, and goes to the points 1, 2, 3, 4 and 5. The end of the trajectory is represented in Fig. 10.67 by the *end* label.

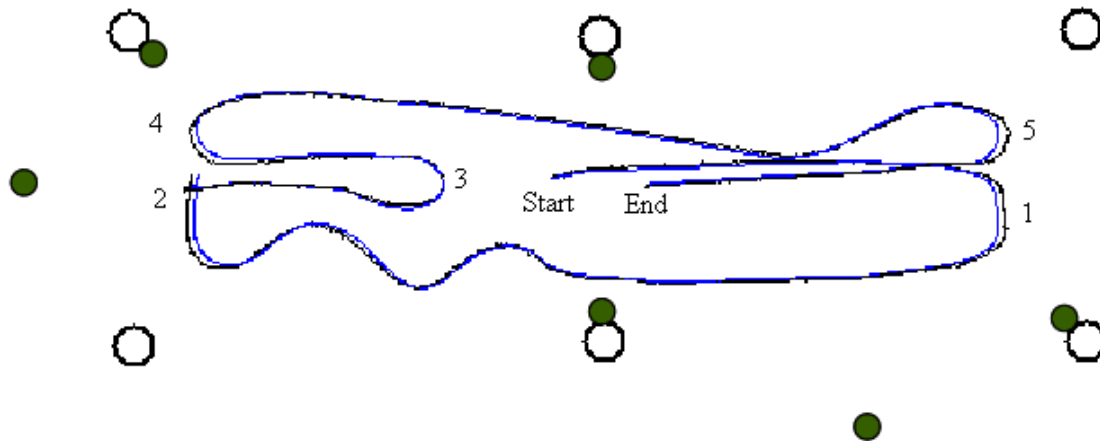


Fig. 10.67. Estimated location compared with the Sick Nav350. The black line is representative of the Nav350 self location. The blue vehicle trajectory is the estimated location. The green six circles are the reflectors. Area of 25 metres of length and 8 metres of width.

In the Fig. 10.67, the green circles are representative of the reflectors used by the location sensor Nav350. The black circumferences are the columns in the scenario.

The following table, Table 10.6, shows the error of localization between the 3D matching algorithm and the Nav350 positioning system.

$\varepsilon_{Nav350}^{3DMatching}$	Localization			
	$\varepsilon_{x_v} (metres)$	$\varepsilon_{y_v} (metres)$	$\varepsilon_{\theta_v} (radians)$	$\varepsilon_d (metres)$
$\hat{\varepsilon} = 1/N \sum \varepsilon$	0.0062	-0.0057	0.0106	0.0847
$\sigma_{\varepsilon} = 1/N \sum \varepsilon^2$	0.087	0.036	0.024	0.091
$\hat{\varepsilon} + 2 \cdot \sigma_{\varepsilon}$	[-0.17, 0.18]	[-0.077, 0.066]	[-0.037, 0.059]	[0, 0.27]

Table 10.6 Results of the average and standard deviation of  $\varepsilon_d$  and  $\varepsilon_{\theta}$ . Comparison between the 3D matching the Nav350 localization positioning system.

With the table analysis, it can be concluded that the distance and angle errors, between the two systems of localization, are into the intervals [0, 0.27] metres and [-0.037, 0.059] radians, with a certainty of 95.4% (two times the standard deviation).

The combination of the Table 10.3 and Table 10.6 generates the Table 10.7. This last table has the errors and standard deviations between the localization system proposed in this thesis, and the ground truth. The Table 10.7 results from the equations (10.10) and (10.11).

$\varepsilon_{GND}^{3DMatching}$	Localization			
	$\varepsilon_{x_v} (metres)$	$\varepsilon_{y_v} (metres)$	$\varepsilon_{\theta_v} (radians)$	$\varepsilon_d (metres)$
$\hat{\varepsilon} = 1/N \sum \varepsilon$	0.00614	-0.0031	0.0136	0.0957
$\sigma_{\varepsilon} = 1/N \sum \varepsilon^2$	0.087	0.039	0.024	0.093
$\hat{\varepsilon} + 2 \cdot \sigma_{\varepsilon}$	[-0.17, 0.18]	[-0.081, 0.075]	[-0.035, 0.062]	[0, 0.28]

Table 10.7 Results of the average and standard deviation of  $\varepsilon_d$  and  $\varepsilon_{\theta}$ . Comparison between the 3D matching and GND.

Finally, with the table analysis, it can be concluded that the 3D matching errors  $\varepsilon_{\theta_v}$  and  $\varepsilon_d$ , are in the intervals [-0.035, 0.062] radians and [0, 0.28] metres, in the 95.4% of the cases.

The Fig. 10.68 to Fig. 10.70 are demonstrative of the localization variables (3D matching in red and the estimation given with the Nav350 localization system in blue). The Fig. 10.71 presents the error between the two localization systems. The x position error is ever lower than 20 centimetres. The y position error is ever lower than 15 centimetres, while the  $\theta$  error is in almost the times lower than 0.1 radians.

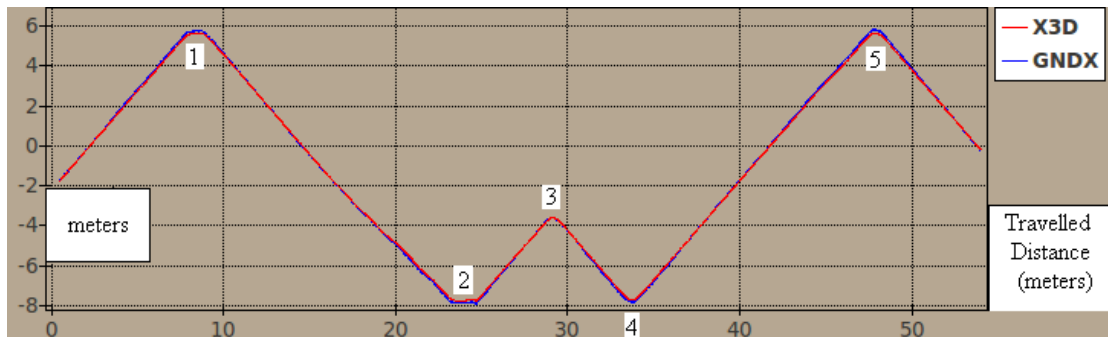


Fig. 10.68 x variable in function to the travelled distance. The vehicle estimated  $x_v$  is represented in red, while in blue is shown the Nav350's  $x$  value.

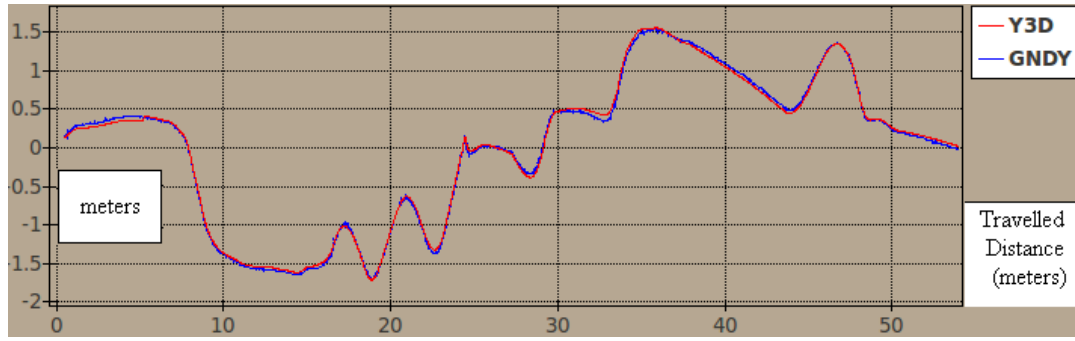


Fig. 10.69  $y$  variable in function to the travelled distance. The vehicle estimated position  $y_v$  is represented in red, while in blue is shown the Nav350's  $y$  position.

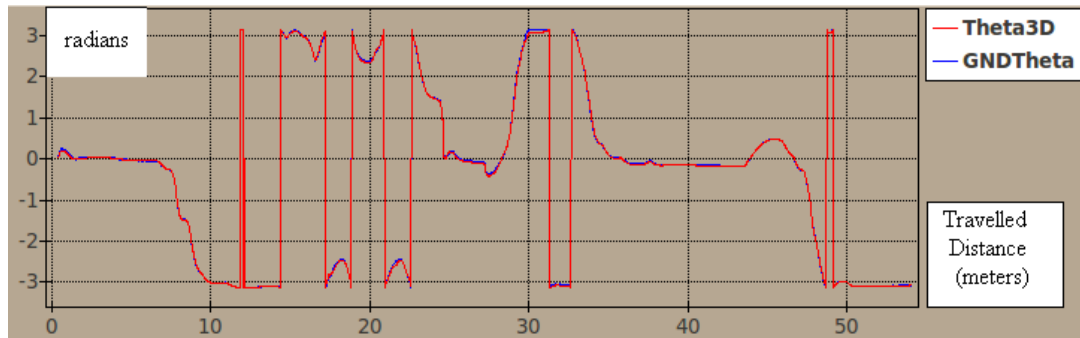


Fig. 10.70  $\theta$  variable in function to the travelled distance. The vehicle estimated position  $\theta_v$  is represented in red, while in blue is shown the Nav350's  $\theta$  position.

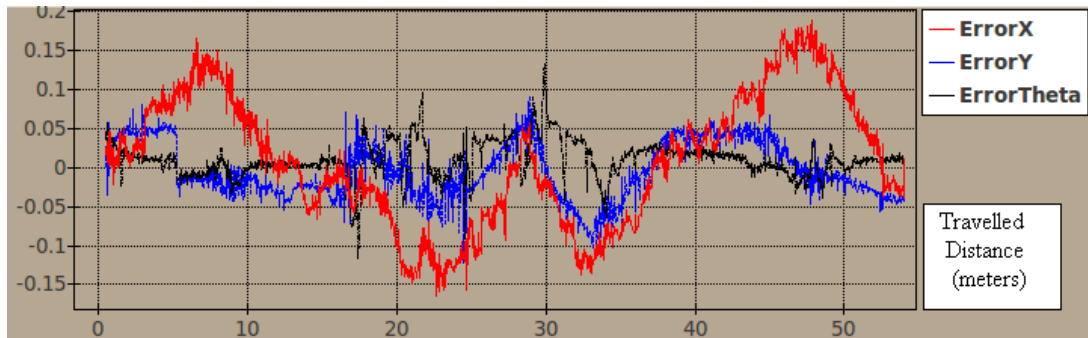


Fig. 10.71 Error curves between the 3D matching localization and the Nav350 positioning system estimation.

Finally, the results of accuracy of the third scenario, were also obtained, comparing the 3D matching localization algorithm of the *three-dimensional map-based* approach and the industrial positioning system Nav350.

As can be seen in Fig. 10.72, the localization position estimation (blue line) is close to that obtained with Nav350 self location (black line). The trajectory begins in the *start* and goes to the point *1*, *2* and ends in the *end*, as shown in Fig. 10.72.

It is shown, in the Table 10.8, the error of the 3D matching localization algorithm when compared with the Nav350 location as reference. The table helps to realize that the distance and angle errors,  $\varepsilon_d$  and  $\varepsilon_{\theta_v}$ , considered Gaussian errors, are, in this scenario, with a certainty of 95.4%, inside the intervals  $[0, 0.19]$  metres and  $[-0.059, 0.072]$  radians, respectively.

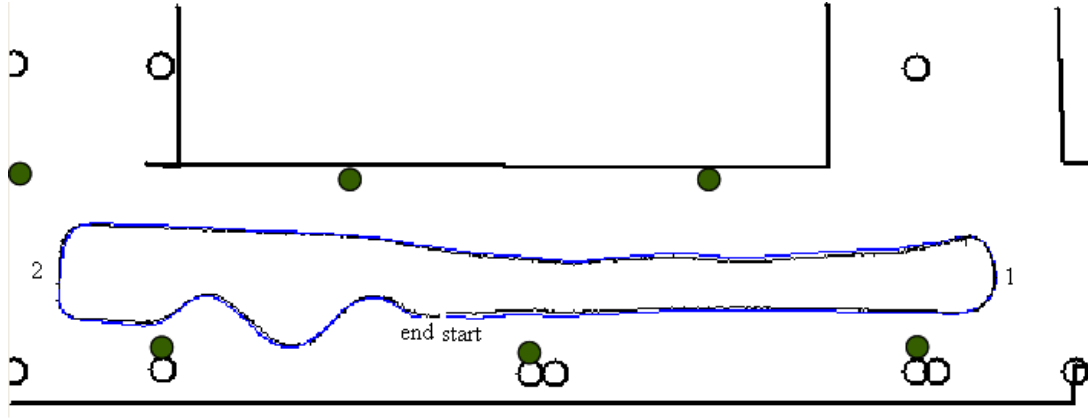


Fig. 10.72. Estimated location with reference with the Sick Nav350 position. The black vehicle position is representative of the Nav350 self location. The blue vehicle trajectory is the estimated location. The green six circles are reflectors placed by the scenario. The corridor has 25 metres long and 8 metres wide.

$\varepsilon_{Nav350}^{3DMatching}$	Localization			
	$\varepsilon_{x_v} (metres)$	$\varepsilon_{y_v} (metres)$	$\varepsilon_{\theta_v} (radians)$	$\varepsilon_d (metres)$
$\hat{\varepsilon} = 1/N \sum \varepsilon$	0.0047	0.0032	0.0069	0.0634
$\sigma_{\varepsilon} = 1/N \sum \varepsilon^2$	0.041	0.053	0.033	0.066
$\hat{\varepsilon} + 2 \cdot \sigma_{\varepsilon}$	[-0.077, 0.087]	[-0.10, 0.11]	[-0.059, 0.072]	[0, 0.19]

Table 10.8 Results of the average and standard deviation of  $\varepsilon_d$  and  $\varepsilon_{\theta}$ . Comparison between the 3D matching the Nav350 localization positioning system.

Considering, as already referred, the error of the Nav350 positioning system related with the ground truth has a Gaussian distribution. Analogy, the error between the 3D matching localization and the Nav350 sensor is considered Gaussian. Therefore, the tables, Table 10.3 and Table 10.8, when combined as shown by the equations (10.10) and (10.11), generate the Table 10.9.

With the Table 10.9 analysis it is possible to conclude that the errors of distance and angle,  $\varepsilon_d$  and  $\varepsilon_{\theta_v}$ , respectively, are inside the intervals of [0, 0.21] and [-0.057, 0.076] radians, with a certainty of 95.4% (equivalent to  $\hat{\varepsilon} \mp 2 \cdot \sigma_{\varepsilon}$ ).

$\varepsilon_{GND}^{3DMatching}$	Localization			
	$\varepsilon_{x_v} (metres)$	$\varepsilon_{y_v} (metres)$	$\varepsilon_{\theta_v} (radians)$	$\varepsilon_d (metres)$
$\hat{\varepsilon} = 1/N \sum \varepsilon$	0.0046	0.0058	0.0099	0.074
$\sigma_{\varepsilon} = 1/N \sum \varepsilon^2$	0.041	0.055	0.033	0.068
$\hat{\varepsilon} + 2 \cdot \sigma_{\varepsilon}$	[-0.078, 0.088]	[-0.10, 0.12]	[-0.057, 0.076]	[0, 0.21]

Table 10.9 Results of the average and standard deviation of  $\varepsilon_d$  and  $\varepsilon_{\theta}$ . Comparison between the 3D matching and GND.

The following figures, Fig. 10.73 to Fig. 10.75 are demonstrative of the localisation variables (3D matching at red and the estimation given with the Nav350 localization system in blue). The Fig. 10.76 presents the error between the two localization systems. The x

position error is ever lower than 15 centimetres. The  $y$  position error is ever lower than 15 centimetres, while the  $\theta$  error is in almost the times lower than 0.1 radians.

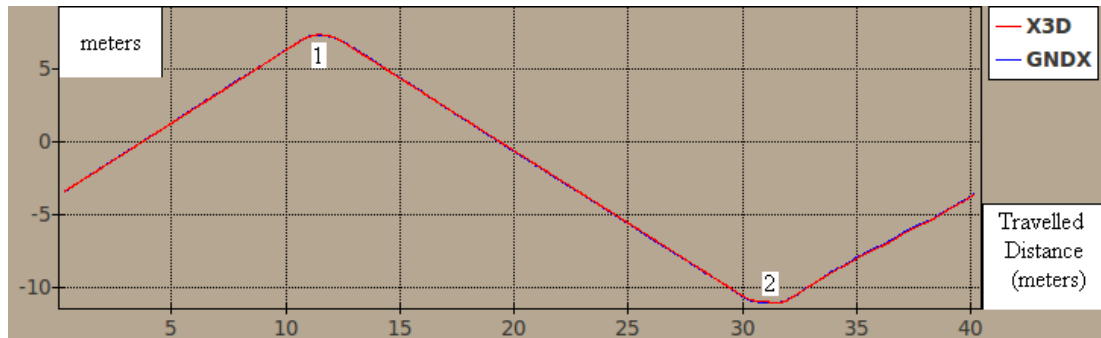


Fig. 10.73  $x$  variable in function to the travelled distance. The vehicle estimated position  $x_v$  is represented in red, while in blue is shown the Nav350's  $x$  position.

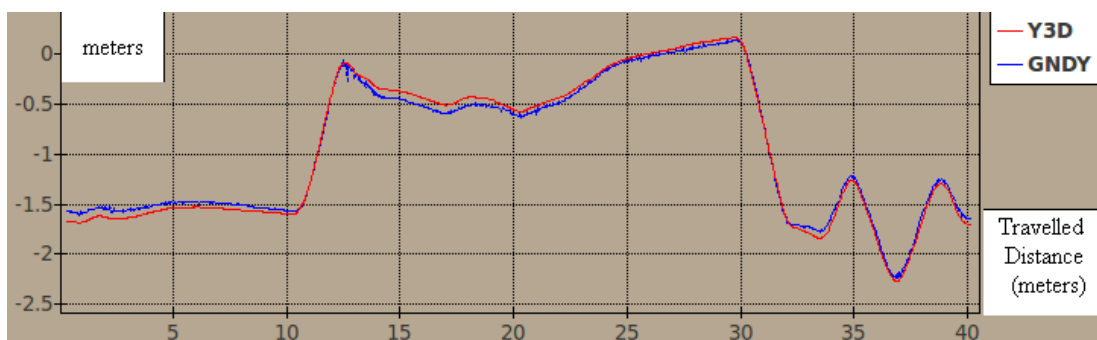


Fig. 10.74  $y$  variable in function to the travelled distance. The vehicle estimated position  $y_v$  is represented in red, while in blue is shown the Nav350's  $y$  position.

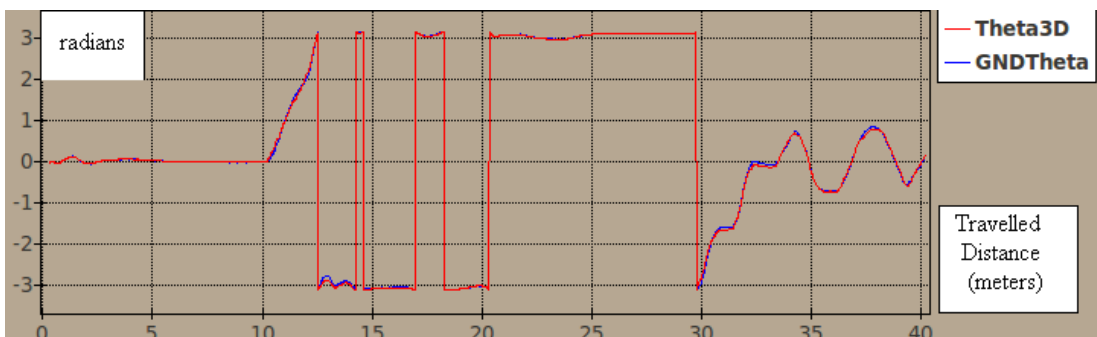


Fig. 10.75  $\theta$  variable in function to the travelled distance. The vehicle estimated position  $\theta_v$  is represented in red, while in blue is shown the Nav350's  $\theta$  position.

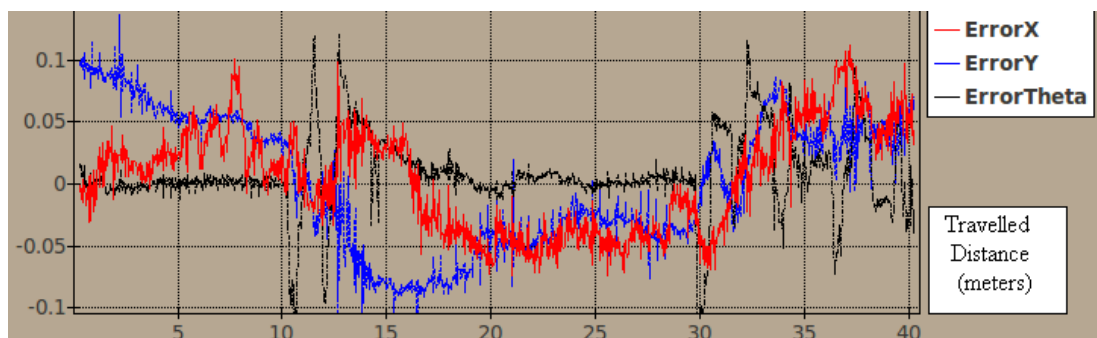


Fig. 10.76 Error curves between the 3D matching localization and the Nav350 positioning system estimation.

With the results presented in this Chapter, about the localization accuracy in the three scenarios already described, it is possible to conclude that the distance error ( $\varepsilon_d$ ) is lower than



0.31 metres in its worst case with a confidence of 95.4%, see Table 10.5. The angle error, equal to  $\varepsilon_{\theta_v}$ , is in the worst case lower than 0.076 radians, with a certainty of 95.4% (two times the standard deviation), see Table 10.9.

The 3D matching localization accuracy varies according to the scenario features and structure. The three scenarios, although different, are well-structured scenarios. In these environments the localization based in 3D matching is sufficiently accurate to be used to pinpoint the vehicle localization during autonomous surveillance tasks, as is possible to see in videos at the webpage [3].

The performed experiments helped to realize as well some advantages of the proposed algorithm of localization, when compared with the Sick Nav350 positioning system. These advantages are: the observation module as described in Chapter 5, *Hokuyo URG-04LX-UG01* and *AX-12 Dynamixel Bioloid*, has a lower cost, is considerably lighter and, becoming it more suitable to be used in smaller and service robots as is the case of the RobVigil.

### 10.4.3. Robustness of the Proposed Methodology

Two crucial experiments were conducted in order to justify the advantages of: 1) using 3D Matching during the vehicle's normal operation comparatively to 2D Matching; 2) using a rotating LRF, instead a fixed LRF (in a vertical position) as observation module. The two experiments are described as follows:

1) The 2D Matching algorithm was executed using the fixed observation module (LRF horizontally) and it was verified that it is not robust when dynamic objects appear.

2) The advantage of using a rotating observation module was verified. The 3D Matching algorithm was executed using the fixed observation module (LRF vertically). The results were then compared with those obtained using the rotating LRF.

Fig. 10.77 is representative of experiment 1). Therefore, Fig. 10.77 shows the matching performed by the 2D matching algorithm, with the fixed observation module (LRF horizontally). The black lines are the walls where the matching should be performed. The blue points represent the observation module points.

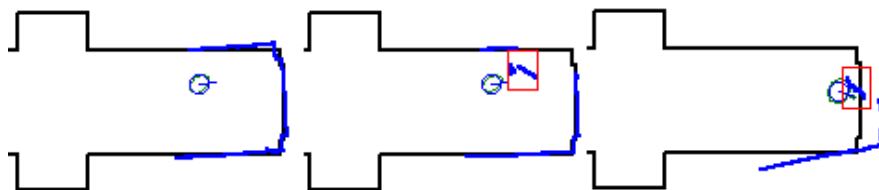


Fig. 10.77 2D Matching with a dynamic object appearing (red rectangle). Image on the left: without a dynamic object. Image in the middle: with a person in front of the robot. Image on the right: final matching between the actual reading and the 2D Map. Matching is wrong due to the dynamic object.

In the figure is shown an initial situation (image on the left) where matching is performed correctly. The image in the middle demonstrates a dynamic object appearing in the scenario, represented by the red box. The image on the right represents a final result after the appearance of a dynamic object.

Throughout experiment 1) it was possible to conclude that 2D Matching is not robust when dynamic objects appear in the scene. As a result, matching is not correct and that is confirmed in Fig. 10.77 (image on the right).

On the other hand, as with the 3D matching algorithm only the upper side of the indoor environment is used (above 1.8 metres ), because dynamic objects are not common in that area, it operates correctly even in scenes with lot of people travelling.

However, the 3D Matching algorithm is robust even when there are dynamic objects on the upper side of the indoor scenario. Imagine a tall object or person, more than 1.8 metres of high, crossing the vehicle's navigation area. As previously described in Chapter 5, the observation module is an LRF placed in a rotating platform which makes it possible to obtain information on directions that are different from the directions in which the dynamic object appeared.

Therefore, the rotating capacity of the observation module allows the 3D matching algorithm to perform the matching process in different areas of the map which are predominantly static. As a result, the localization process becomes more robust even when the vehicle navigates in dynamic scenarios.

Fig. 10.78 and Fig. 10.79 are representative of experiment 2). Both figures demonstrate the performance of the 3D matching algorithm. However, Fig. 10.78 shows the matching result with the observation module placed vertically (LRF vertically). On the contrary, Fig. 10.79 shows when the observation module is rotating. The blue points represent the observation module points used in the localization method, projected in the 2D space (points  $P_{3D}$  projected in the 2D space).



Fig. 10.78. Experiment conducted with a fixed observation module (LRF vertically). Image on the left: The performed 3D matching. Image on the right: The inverse of the quadratic matching error.

Fig. 10.79. Experiment conducted with the observation module rotating. Image on the left: 3D Matching. Image on the right: The inverse of the quadratic matching.

The images on the right in Fig. 10.78 and Fig. 10.79 represent the inverse of the quadratic matching error, when the vehicle is in any position of the area represented in the figures. The lighter areas (red boxes) represent those places with lower matching errors. Therefore, these areas represent local minima of the cost function.

In the experiment conducted and depicted in Fig. 10.78, the estimation of the  $y_v$  position is reliable as opposite to the  $x_v$  position (parallel to the corridor walls) and  $\theta_v$  angle. In this situation, there are plenty of local minima (red box) of the cost function along the  $x_v$  axis, which makes it more difficult to estimate the  $x_v$  and  $\theta_v$  values.

In the experiment depicted in Fig. 10.79, estimating the positions of  $x_v$ ,  $y_v$  and  $\theta_v$  is equally reliable. In this situation, there is only a distinctive local minimum (red circle) of the cost function (see Fig. 10.79 right-hand side).

Throughout experiment 2), it was possible to conclude that using a vertically fixed module provides a less reliable matching result in the direction parallel to the walls.

The use of a rotating module provides a matching result that is equally reliable in all directions ( $x_v$ ,  $y_v$  and  $\theta_v$ ). This happens due the fact that the rotating observation module has

the ability to acquire more quality and different information which is helpful when estimating all the variables  $x_v$ ,  $y_v$  and  $\theta_v$ .

In summary, with regard to 1) and 2), it is possible to say that: 1) the 2D Matching is not a good approach to locating the vehicle when the scenario is not static; 2) the 3D Matching algorithm is preferable due its feasibility and robustness when the scenario used is the upper side of a building (the more static scenario); 3) the 3D Matching should be conducted while the observation module is rotating (not fixed vertically) since the collected information is more helpful to estimate the vehicle's position, which makes it a more reliable technique.

#### 10.4.4. Execution Time and Comparison with Other Algorithms

Regarding the *three-dimensional map-based* approach, the localization algorithm described in this document, which is based on 3D matching, it has a consumption time whose maximum value depends on the number of points used in the matching algorithm. As the number of points acquired by the LRF is limited to 682 points, the maximum time spent in the localization algorithm is also limited. This limit time was measured and is equal to 17 milliseconds. The average execution time is about 12 milliseconds. Both the maximum and mean time spent, are lower than the sample rate imposed by the observation module (100 milliseconds), which allows the algorithm to be used online, with three-dimensional data, in the Mini ITX, EPIA M10000G with a processor of 1.0GHz.

In the *three-dimensional map-based* approach the algorithm Perfect Match described by M. Lauer *et al.* [4], was adapted to operate in the three-dimensional space, with Laser Range Finder data, instead the use of artificial vision. The computational complexity and requirements of the Perfect Match was maintained, even using three-dimensional data.

Since the computation time spent is not compromised, the experiment conducted by M. Lauer *et al.* [4], which elects the Perfect Match as a faster algorithm when compared with the Particle Filter algorithm, remains valid. In this experiment, while the Perfect Match has a spent time of 4.2 milliseconds, the Particle Filter, using 200 and 500 particles, takes 17.9 milliseconds (four times higher) and 48.3 milliseconds (ten times higher), respectively.

Furthermore, the computational time spent (about 2 milliseconds in the 5DPO robots) with the artificial vision's image treatment was improved, since the used data, is acquired by the developed *observation module*, whose treatment consists in the simpler application of homogeneous transformations in the LRF's raw points, as described in Chapter 5.

Finally, comparing with the localization method described by M. Lauer *et al.* [4], the *localization* procedure of *three-dimensional map-based* approach, described in this PhD thesis, was improved. It was applied the Extended Kalman Filter as multi fusion sensor system, aiming to join the odometry information and the three-dimensional Perfect Match result.

Comparing the localization algorithm proposed and described in this PhD thesis with the ICP algorithms, it is faster and can be applied online in smaller cycle times, even when using a larger quantity of Laser Range Finder points.

The MbICP algorithm described in [38], which already shows improvements to the standard ICP, takes an average time of 76 milliseconds to find the optimal solution, using only 361 Laser Range Finder points, in a sample rate of 200 milliseconds, in a Pentium IV 1.8GHz. Furthermore, the approach proposed in this PhD thesis, has a limited time of

execution, depending on the number of the used points. In the ICP algorithms, the point to point match step is necessary to obtain the correspondence with the previous scan. Such step is not limited in the execution time and is widely dependent on the overlapping zone between consecutive scans. This overlapping zone influences also the quality of the reached solution.

The use of three-dimensional data on the upper side of a building, increases the quantity and quality of the information, especially because it is almost static.

Until the moment, the author did not find other works in the state of the art, which enter in account with the following considerations: the use of three-dimensional data together with a 3D map for the online localization of a robot; using only data on the static scenario of a building, making the localization more robust, even when navigating inside dynamic scenarios.

Also it was not found other works where the SLAM problem in 2D space was solved with the intention of mapping the environment in the 3D space, using for that the same LRF (tilting LRF). Furthermore, any work was found, where the 3D mapping purpose is to use the resultant occupancy grid to localize a robot using three-dimensional data.

Between the state of art works that were read by the author, which addresses the topic of SLAM, anyone uses data about invariant points from corners and columns simultaneously with linear segments.

In contrast to the work described by S. Thrun *et al.* [46], in the *pre-localization* and *mapping*, only one Laser Range Finder (observation module) is used in this work. The single observation module is used to perform the *pre-localization* and *mapping* routines simultaneously. Furthermore, after, the map constructed in the 3D space is used locate the vehicle in its normal operation.

Contrary to the work described in [47], the matching algorithm presented to pinpoint the location of a robot uses 3D data, enabling the three-dimensional scan to align with the previous map stored in memory.

T. Yokoya *et al.* in [48] uses more than one robot and sensorial units to perform the three-dimensional mapping of the surrounding. In this work, an approach was developed, which allows to: 1) locate the vehicle in a 2D space, in indoor environments; 2) map the surrounding environment, representing it in 3D models; 3) and locate the vehicle in a shorter amount of time, during its regular operation, enabling its online application.

K. Nishimoto *et al.* [49] uses detected plans and the respective extracted information, through the Hough Transform, to pinpoint the vehicle position. Unfortunately, in an experiment the calculation of the Hough Transform in Celeron 2.3 GHz, takes a minimum time required of 45 seconds, becoming non-applicable when online performance is required.

With the revision of literature it is possible to conclude the following:

1) A lot of the work focuses on the topic of 2D SLAM in indoor environments. However, some drawbacks appear: non static environments are found and the SLAM solutions are computationally heavy, inapplicable online.

2) The Iterative Closest Point algorithms are popular in the literature due to the simplicity of their operation, but they fail in the computing time.

3) There is not an entire localization methodology, that treats the mapping and posterior localization of the vehicle in the three-dimensional space.

4) Furthermore, the found methodologies don't devote themselves, neither to problems about time spent in localization, the online applicability and the computational requirements, when 3D data are used; nor with the disturbance introduced in the acquired data by persons or objects travelling in dynamic scenarios.

Finally experiments were made, aiming to compare the *three-dimensional map-based* approach and the most used algorithms for mapping (*gmapping* [76]) and localization (*amcl* [77]) from the platform ROS (Robot Operating System).

The ROS platform provide a package of mapping called *gmapping* [76], which is based on the Rao-Blackwellized Particle Filter. Using this package, a 2D grid map was obtained as shown in Fig. 10.80. This grid map is about the corridor where, as presented in the previous sub-section, the first test of accuracy of the 3D Matching algorithm was made.



Fig. 10.80. 2D map grid obtained using the *gmapping* package of ROS.

Another ROS' package, the *amcl*, the adaptative Monte Carlo localization algorithm, uses the 2D map grid built by the *gmapping* package to locate the vehicle, using 2D data, with a Laser Range Finder on the horizontal. In contrary, the approach presented here, the *three-dimensional map-based*, uses the 3D map built during the *pre-localization* and *mapping*, and three-dimensional data acquired by a tilting LRF, to pinpoint the RobVigil pose online.

The feature map obtained by the EKF-SLAM, about the same corridor is presented in Fig. 10.7, while the correspondent 3D occupancy grid, is shown in Fig. 10.25. A slice about the distance and gradients matrices, corresponding to this 3D occupancy grid, are shown in Fig. 10.36 to Fig. 10.38.

An experiment was made, aiming to compare the execution time of the 3D Matching algorithm proposed here and the localization provided by *amcl* package. Also an experiment in the corridor defined above, was made aiming to compare the accuracy of the entire methodology *three-dimensional map-based* (*pre-localization*, *mapping* and *localization*) and the entire group *gmapping* and *amcl* packages.

To perform these experiments, the same data of odometry and the same Laser Range Finder, the Hokuyo-URG-04LX-UG01, was used for both. The results were obtained in the same vehicle trajectory, executed in the same computer, the RobVigil's onboard pc (Mini ITX, EPIA M10000G with a processor of 1.0GHz). In the tests of accuracy, also the Nav350, was used as ground truth, such as described in the previous sub-section.

The graphic represented in the following figure, Fig. 10.81, shows during 894 iterations the execution time of the 3D Matching algorithm at dashed line. In this graphic it is possible to see the average value at continuous blue line, equal to 11.8 milliseconds, with a maximum value of 17 milliseconds.

The following graphic also shows, the execution time of the localization provided by the *amcl* package. With 1000 particles, the maximum execution time of the *amcl* package is 42

milliseconds. The average value of the execution time is 31.8 milliseconds. The execution time of the *amcl* package using 1000 particles, is higher when compared with the 3D Matching algorithm. Using 500 particles, the average execution time of the *amcl* package is similar to the 3D Matching algorithm proposed in this thesis.

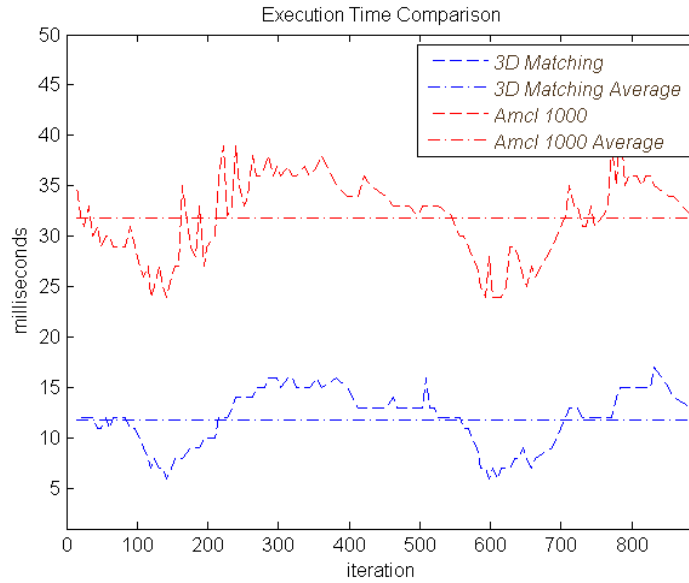


Fig. 10.81. Execution time of the *localization* phase (3D Matching algorithm).

In the test of accuracy, the 3D Matching algorithm, reached similar results to the ones, presented in the Table 10.5. The average angle error is equal to 0.003 radians and average Euclidean distance error is 0.12 metres. The error of the angle and distance are with a certainty of 95.4% (two times de standard deviation), inside the intervals  $[-0.05, 0.062]$  radians and  $[0, 0.29]$  metres, respectively.

The following table shows the results on the accuracy about the angle and distance errors, for the *amcl* package using 1000 and 500 particles. The obtained accuracy results proved that the approach of *three-dimensional map-based* is more accurate when compared to the operation of the group *gmapping* and *amcl*. For the best case, when using 1000 particles on the *amcl*, the average angle error is 2.3 times higher than the *three-dimensional map-based* approach proposed in this thesis, while the average error on the distance is 1.7 times higher.

	AMCL with 1000 particles		AMCL with 500 particles	
	$\varepsilon_{\theta_v}$ (radians)	$\varepsilon_d$ (meters)	$\varepsilon_{\theta_v}$ (radians)	$\varepsilon_d$ (meters)
$\hat{\varepsilon} = 1/N \sum \varepsilon$	0.0073	0.209	0.012	0.221
$\hat{\varepsilon} + 2 \cdot [-\sigma_{\varepsilon}, \sigma_{\varepsilon}]$	$[-0.05, 0.065]$	$[0.046, 0.34]$	$[-0.058, 0.084]$	$[0.036, 0.408]$

Table 10.10 Results of the average and standard deviation of  $\varepsilon_d$  and  $\varepsilon_{\theta}$  for the *amcl* package of ROS running with 500 and 100 particles.

Furthermore, the *three-dimensional map-based* approach, proposed here, uses three-dimensional data about the upper side of the map, the static scenario, to compute its pose in a more robust way, even when applied in dynamic scenarios.

# Conclusions

A first and simpler localization approach has been developed, the *simple landmark-based localization* approach, which is a strategy of localization based on the walls of a square-shaped room. This localization approach is constrained in terms of applicability and in terms of the vehicle navigation area. This approach was developed in a Lego NXT in a real and a simulation scenario, in a suitable way, allowing the applicability of this tool in the lessons of Autonomous Robotic Systems course at the Faculty of Engineering of Porto, so as to teach mobile robot concepts to undergraduate students.

With regard to the *simple landmark-based localization* approach, some conclusions were obtained. This strategy of localization lacks in its applicability in different scenarios, mainly when is required an approach capable of performing its function without constraints in terms of navigation area.

However, this localization approach, proved that it is possible to perform vehicle location using natural landmarks, with no need to prepare the environment.

Another conclusion reached was that a localization approach like that, developed in a suitable robotic platform, as is the example of the Lego NXT, can be really helpful to teach and improve the undergraduate students' learning on the topic of mobile robots.

The *simple landmark-based localization* approach resulted in a paper on an International Journal [68].

After, this thesis evolved to another approach, with a realistic applicability, without constraints in terms of navigation applicable to any indoor well-structured environment, this approach was called *three-dimensional map-based*.

In this document, the entire and new *three-dimensional map-based* methodology of localization is proposed. This methodology features two crucial stages: firstly the setup *preparation task* (*pre-localization* and *mapping*) is performed, which enables the second, pinpointing the location of a vehicle online during the vehicle normal operation (*localization*).

A tilting LRF solution was developed, called *observation module*, based on a low cost 2D LRF and a cheaper dc servo motor. This tilting LRF allows obtaining data in the three-dimensional space. With a set of homogeneous operations it is possible to transform the distance measurements of the LRF into a point on the three-dimensional space.

The *pre-localization* and *mapping* procedures are performed offline, with odometry and data from the *observation module*, logged when the vehicle is controlled with a joystick in a controlled scenario (without people or dynamic objects travelling).

The *pre-localization* uses the EKF-SLAM solution to obtain a feature map, with linear segments and invariant points. Still, during *pre-localization*, the EKF-Loc stage is used to compute the vehicle pose, by using the previously obtained feature map.

Therefore, the *pre-localization* works as an indoor "GPS", allowing to know the vehicle position, to be used during the *mapping* procedure. Through the knowledge on the vehicle pose, and by applying the Bayes Rules, the probability and occupancy grid of the surrounding environment is constructed in the three-dimensional space.

The EKF-SLAM algorithm builds a feature map, which contains not only linear segments representing walls, doors or furniture, but also invariant points, representing columns and corners. This redundancy of information allows an improvement on the EKF-SLAM and EKF-Loc accuracy.

The three-dimensional occupancy grid built allows creating, as well in the three-dimensional space, the look-up tables: distance and gradient matrices. These are inherited by the *localization* procedure from the *pre-localization* and *mapping* stage.

Therefore, the *localization* procedure is a 3D Matching algorithm, which performs a matching in the three-dimensional space, using data acquired by the *observation module* and the 3D map known a priori, to compute the vehicle pose.

Some important conclusions can be presented with regard to the *three-dimensional map-based* approach, as described in the following paragraphs.

The distance and gradient matrices are the core of the *localization* algorithm, since they are look-up tables whose content is pre-computed and used during the 3D Perfect Match algorithm. These matrices become the *localization* algorithm fast enough (average time of 12 milliseconds), using 682 points of the tilting LRF, requiring low computational power. Therefore, this can be executed online in a low power processor, such as the Mini ITX, EPIA M10000G with a 1.0GHz processor.

As the developed 3D Matching uses three-dimensional data, the upper side of the indoor environment (map above the height of a normal person - 1.8 metres - headroom) can be used to performed the vehicle localization. The headroom of a building does not have furniture or people/dynamic objects travelling, thus remaining the same during long periods of time. That way, it can be called as a static scenario, making of the localization method described here, an algorithm with benefits in terms of robustness and feasibility. The *three-dimensional map-based* approach works in well-structured mapped environments, with no need to prepare the environment with artificial landmarks.

The entire set of tests and public demonstrations (a total of eight public demonstrations), performed in public facilities, helped to realize that this localization method proved to be sufficiently robust, even when applied in dynamic scenarios. The *three-dimensional map-based* approach allowed vehicle navigation during interruptible and autonomous surveillance routines, in these demonstrations, even with lot of curious people around the robot.

The observation module used is based on the LRF. However, any observation module capable of acquiring three-dimensional data can be used, such as the Kinect, the 3D camera (MESA), stereoscope vision or commercial 3D LRFs.

It is also possible to conclude that, with the conducted experiments to characterize the accuracy of the *localization*, the error of the estimation on the pose of the vehicle, in relation to the ground truth is negligible, mainly when we consider the accuracy requirements required for the RobVigil navigation.



The developed localization methodology makes the RobVigil a cost-effective practicable platform, since its single cost is the tilting LRF (1000 Euros is the cost of the LRF Hokuyo and 100 Euros is the cost of the DC Servo motor).

The *three-dimensional map-based* approach resulted in three publications, two of them in conferences, [70] and [71], and the other one in an International Journal, [72]. The entire project had a lot of impact and projection in the Portuguese press. Some videos on press news and newspaper's covers, related to the RobVigil, can be seen on the webpage [3].

Finally, this thesis was integrated as a component of the project: "QREN-SI à Investigação e Desenvolvimento Tecnológico, projecto de I&DT Empresas em co-promoção - "RobotVigilante".



## Future Work

In the future, this work can evolve in three different topics and research lines. The first line of research is related with the rotation of the tilting Laser Range Finder. This task of rotation, is done with the fundamental goal of acquire more quantity and helpful information allowing a more accurate and robust localization. But such rotation, leads to an increase of the energy consumption due the servo motor's operation. Besides that, the servo motor's rotation leads to its own mechanical degradation. Therefore, ideally the tilting Laser Range Finder rotation should enter in account with the localization needs. The tilting LRF should rotate only when it is necessary to find zones where the acquired information is helpful for the vehicle location.

The second line of research is related with the initial position of the vehicle. At the moment, in the methodology proposed on the second approach, the *three-dimensional map-based*, the robot as no idea on its initial location. The initial position of the vehicle is given as a parameter, or otherwise, the vehicle can start a task or routine in the same initial and known position (like a docking station). In future, the task of pinpoint the initial position of the vehicle, allowing it to start in any initial pose, autonomously, can be studied and developed. In fact this work was already started, with some preliminary results, which were published on the paper [70].

Finally, the third line of research is related with the capacity of the localization algorithm to estimate also the vehicle's z coordinate, roll and pitch angles. In that way, will be possible the vehicle navigation through irregular floors, which includes inclined surfaces as is example ramps. A possible approach is the use of an IMU as sensor to perform dead-reckoning, instead or complementing the odometry, and the adaptation of the 3D Matching algorithm to estimate those state variables as well (vehicle's z coordinate, roll and pitch angles).



# References

- [1] D. Schoenwald, "Autonomous Unmanned Vehicles: In Space, Air, Water, and on the Ground", IEEE Control Systems, Vol. 20, No. 6, pp. 15-18, December 2000.
- [2] M. Pinto, "Localization of Mobile Robots Using an Extended Kalman Filter in a Lego NXT". Available at "[www.fe.up.pt/~miguelp/LegoNxt](http://www.fe.up.pt/~miguelp/LegoNxt)", June 2012.
- [3] M. Pinto, "SLAM for 3D Map building to be used in a 3D Matching Localization Algorithm". Available at "[www.fe.up.pt/~dee09013](http://www.fe.up.pt/~dee09013)", June 2012.
- [4] M. Lauer, S. Lange and M. Riedmiller, "Calculating the perfect match: an efficient and accurate approach for robot self-localization", RoboCup Symposium, pp. 142-53, Osaka, Japan, 13-19 July, 2005.
- [5] Groover, M. P., (2000). Automation, Production Systems, and Computer - Integrated Manufacturing. New Jersey: Prentice-Hall.
- [6] L. Schulze, S. Behling and S. Buhrs, "AGVS in Logistics Systems State of the Art, Applications and New Developments", International Conference On Industrial Logistics (ICIL 2008): Logistics in a Flat World, Strategy, Management and Operations, pp. 256-264, Tel Aviv, Israel, 9-15 March, 2008.
- [7] J. Borenstein, H. R. Everett, L.Feng and D. Wehe, "Mobile Robot Positioning and Sensors and Techniques", Journal of Robotic Systems, Special Issue on Mobile Robots, Vol. 14 No. 4, pp. 231-249, April 1997.
- [8] J. Borenstein and L.Feng, "Measurement and Correction of Systematic Odometry Errors in Mobile Robots", IEEE Transactions on Robotics and Automation, Vol. 12, No. 6, pp. 869-880, December 1996.
- [9] B. Barshan, H. Durrant-Whyte, "Inertial Navigation Systems for Mobile Robots", IEEE transactions on Robotics and Automation, Vol. 11, No. 3, pp. 328-342, June 1995.
- [10] J. S. Esteves, A. Carvalho and C. Couto, "Position and Orientation Errors in Mobile Robot Absolute Self-Localization Using an Improved Version of the Generalized Geometric Triangulation Algorithm", IEEE International Conference on Industrial Technology (ICIT), Orissa, India, 15-17 December, 2006.
- [11] H. Sobreira, A. P. Moreira and J. S. Esteves, "Characterization of Position and Orientation Measurement Uncertainties in a Low-Cost Mobile Platform", 9th Portuguese Conference on Automatic Control, Controlo 2010, Coimbra, Portugal, pp. 635-640, 8-10 September, 2010.
- [12] A. Matos, N. Cruz, "MARES - Navigation, Control and On-board Software", in Underwater Vehicles, ISBN 978-953-7619-49-7, Austria, January 2009.
- [13] "The Cricket Indoor Location System". Available at "<http://cricket.csail.mit.edu/>", June 2012.
- [14] "Society Robots - Sensors - Robot Sonar". Available at "[http://www.societyofrobots.com/sensors\\_sonar.shtml](http://www.societyofrobots.com/sensors_sonar.shtml)", June 2012.
- [15] "Society Robots - Sensors - IR Range Finder". Available at "[http://www.societyofrobots.com/sensors\\_sharpirrange.shtml](http://www.societyofrobots.com/sensors_sharpirrange.shtml)", June 2012.
- [16] Programming-Computer Vision Tutorial. Society of Robots. Available at: "[http://www.societyofrobots.com/programming\\_computer\\_vision\\_tutorial.shtml](http://www.societyofrobots.com/programming_computer_vision_tutorial.shtml)", August, 2011.
- [17] "Groundsys Group – 5DPO Team". Available at "<http://gnomo.fe.up.pt/~robotic/>", June 2012.
- [18] "Society of Robots - Sensors - Laser Range Finder". Available at "[www.acroname.com/robotics/info/articles/laser/laser.html](http://www.acroname.com/robotics/info/articles/laser/laser.html)", June 2012.

- [19] M. Pinto, A. P. Moreira, P. Costa, M. Ferreira; "Accuracy Analysis of a Flexible Manufacturing System for Picking Cork Pieces in a Conveyor Belt", 20th International Conference on Flexible Automation and Intelligent Manufacturing FAIM2010, California-USA, July 12-14, 2010.
- [20] D. Klimentjew, M. Arli, and J. Zhang, "3D Scene Reconstruction Based on a Moving 2D Laser Range Finder for Service-Robots". IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 19-23, Guilin, China, 19 -23 December, 2009.
- [21] P. Dias, M. Matos and V. Santos, "3D Reconstruction of Real World Scenes Using a Low-Cost 3D Range Scanner", Computer-Aided Civil and Infrastructure Engineering Journal, Vol. 21, No. 7, pp. 486-497, October 2006.
- [22] A. Zhang, S. Hu, Y. Chen, H. Liu, F. Yang and J. Liu, "Fast Continuous 360 Degree Color 3D Laser Scanner", International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVII, Beijing, China, 21 November, 2008.
- [23] H. Surmann, K. Lingemann, A. Nüchter, J. Hertzberg, F. Ais and F. Gesellschaft, "Fast Acquiring and Analysis of Three-Dimensional Laser Range Data", Proceedings of the Sixth International Fall Workshop Vision, Modelling, and Visualization (VMV'01), pp. 59–66, Stuttgart, Germany, 21-23 November, 2001.
- [24] P. H. Batavia, S. A. Roth and S. Singh, "Autonomous Coverage Operations In Semi-Structured Outdoor Environments", IEEE/RSJ International Conference on Intelligent Robots and System, Vol. 1, pp. 743-749, Lausanne, Switzerland, September 30-October 4, 2002.
- [25] H. Surmann, K. Lingemann, A. Nüchter and J. Hertzberg, "A 3D Laser Range Finder for Autonomous Mobile Robots", Proceedings of the 32nd International Symposium on Robotics (ISR'01), pp. 153-158, Seoul, Korea, 19-21 April, 2001.
- [26] F. Moosmann, O. Pink and C. Stiller, "Segmentation of 3D Lidar Data in non-flat Urban Environments using a Local Convexity Criterion", 2009 IEEE Intelligent Vehicles Symposium, pp. 215- 220, Xi'an , China, 3-5 June, 2009.
- [27] M. Bosse and R. Zlot, "Continuous 3D Scan-Matching with a Spinning", IEEE International Conference on Robotics and Automation, ICRA 2009, pp. 4312-4319, Kobe, Japan, 12-17 May, 2009.
- [28] R. Pascoal and V. Santos, "Compensation of Azimuthal Distortions on a Free Spinning 2D Laser Range Finder for 3D Data Set Generation", Proceedings of Robótica 2010, Leiria, Portugal, 24 March, 2010.
- [29] P. Dias, G. Campos, V. Santos, R. Casaleiro, R. Seco and B. S. Santos, "3D Reconstruction and Auralization of the 'Painted Dolmen' of Antelas", In Proceedings of the Electronic Imaging (SPIE), Vol. 6805, 6805OY, Three-Dimensional Image Capture and Applications 2008, San Jose, California, USA, 28-29 Jan, 2008.
- [30] A. Sousa, A. P. Moreira and P. Costa, "Multi Hypotheses Navigation for Indoor Cleaning Robots", 3rd International Workshop on Intelligent Robotics (IRobot 2008), pp. 71-82, Lisbon, Portugal, 14 October, 2008.
- [31] A. Sousa, P. Costa, A. P. Moreira and A. S. Carvalho, "Self Localization of an Autonomous Robot: Using an EKF to Merge Odometry and Vision Based Landmarks", IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2005), pp. 227-234, Catania, Italy, 19-22 September, 2005.
- [32] F. Ribeiro, I. Moutinho, N. Pereira, F. Oliveira, J. Fernandes, N. Peixoto, A. Salgado, "High Accuracy Navigation in Unknown Environment Using Adaptive Control", Robocup 2007, RoboCup International Symposium, pp. 312-319, Atlanta, Georgia, USA, 9-10 July, 2007.
- [33] M. Veloso and J. Biswas, "WiFi Localization and Navigation for Autonomous Indoor Mobile Robots", IEEE International Conference on Robotics and Automation (ICRA), pp. 4379-4384, Anchorage, Alaska, USA, 3-7 May, 2010.

- [34] Thrun, S., & Burgard, W., & Fox, D. (2005). Probabilistic Robotics. Cambridge, Massachusetts: The MIT Press.
- [35] Ge, S. S., & Lewis, F. L. (Eds.). (2006). Autonomous Mobile Robots Sensing, Control, Decision, Making and Applications. New York: CRC Press.
- [36] A. Aghamohammadi, B. Jensen and R. Siegwart, "Scan Alignment With Probabilistic Distance Metric", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004), Vol.3, pp. 2191 - 2196, Sendai, Japan, 28 September - 2 October, 2004.
- [37] J. Minguez, F. Lamiroux, and L. Montesano, "Metric-Based Scan Matching Algorithms for Mobile Robot Displacement Estimation", in Proc. of the IEEE International Conference on Robotics and Automation (ICRA), pp. 3557-3563, Barcelona, Spain, 18-22 April, 2005.
- [38] J. Minguez, F. Lamiroux, and L. Montesano, "Metric-Based Iterative Closest Point Scan Matching for Sensor Displacement Estimation", IEEE Transactions On Robotics, October 2006.
- [39] S. Zhang and G. Yan, "An Efficient Planning Method for Mobile Robot Based 3D Digitization", Proceedings of the 8th World Congress on Intelligent Control and Automation (WCICA), pp. 1348 - 1353, Jinan, China, 6-9 July, 2010.
- [40] S. H. Cho and S. Hong, "Map-based Indoor Robot Navigation and Localization Using Laser Range finder", 11th Int. Conf. Control, Automation, Robotics and Vision, pp. 1559-1564, Singapore, 7-10 December, 2010.
- [41] Castellanos, J. A., & Neira, J., & Tardós, J. D., (2006). Map Building and SLAM Algorithms, on Autonomous Mobile Robots, Sensing, Control, Decision, Making and Applications. New York: CRC Press.
- [42] CSorba, M., (1997), Simultaneous Localization and Map Building, Thesis, (PhD), Robotic Research Group Department of Engineering of Oxford, Oxford, England.
- [43] A. Garulli, A. Giannitrapani, A. Rossi, A. Vicino, "Mobile Robot SLAM for Line-Based Environment Representation", 44th IEEE Conference on Decision and Control Decision and Control, 2005 and 2005 European Control Conference, (CDC-ECC '05), pp. 2041 - 2046, Seville, Spain, 12-15 December, 2005.
- [44] L. Teslić, I. Škrjanc and G. Klančar, "Using a LRF Sensor in the Kalman-filtering-based Localization of a Mobile Robot", ISA Transactions (Elsevier), Vol. 49, No. 1, pp. 145-153, January 2010.
- [45] G. Grisetti, C. Stachniss and W. Burgard, "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters", IEEE Transactions on Robotics, Vol. 23, No. 1, pp. 34-46, February 2007.
- [46] Thrun and D. Fox, "A Real-Time Algorithm for Mobile Robot Mapping With Applications to Multi-Robot and 3D Mapping". Best Conference Paper Award, IEEE International Conference on Robotics and Automation. San Francisco, Vol. 1, pp 321-328, 24-28 April 2000.
- [47] D. Hähnel, W. Burgard, and S. Thrun. "Learning compact 3D models of indoor and outdoor environments with a mobile robot". Robotics and Autonomous Systems (Elsevier), Vol. 44, No. 1, pp. 15-27, July 2003.
- [48] T. Yokoya, T. Hasegawa and R. Kurazume, "Autonomously Generating a 3D map of Unknown Environment by Using Mobile Robots Equipped with LRF", IEEE International Conference on Robotics and Biomimetics, pp. 19-23, Guilin, China, 19-23 December, 2009.
- [49] K. Nishimoto, A. Sagami and K. Kaneyama, "Three Dimensional Recognition of Environments for a Mobile Robot using a Laser Range Finder", International Conference on Instrumentation, Control and Information Technology (SICE 2007), pp. 401 - 405, Kagawa, Japan, 17-20 September, 2007.

- [50] S. Jia, H. Yang, X. Li and W. Cui, "Mobile Robot Localization and Mapping Based on Mixed Model", 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE 2010), Vol. 5, pp. 9-14, Chengdu, China, 20-22 August, 2010.
- [51] S. Jia, H. Yang, X. Li and W. Cui, "SLAM for Mobile Robot Based on Interactive GUI", IEEE International Conference on Mechatronics and Automation (ICMA), pp. 1308 - 1313, Xi'an, China, 4-7 August, 2010.
- [52] A. Rusdinar, J. Kim and S. Kim, "Error Pose Correction of Mobile Robot for SLAM Problem using Laser Range Finder Based on Particle Filter", International Conference on Control Automation and Systems (ICCAS), pp. 52-55, Gyeonggi-do, Korea, 27-30 October, 2010.
- [53] Gelb, A., & Kasper Jr., Nash Jr., R. A., & Price C. F., & Sutherland Jr., A. A., (1996). Applied Optimal Estimation. The MIT Press.
- [54] S. Carpin, A. Birk and V. Jucikas, "On map merging", Robotics and Autonomous Systems (Elsevier), Vol. 53, No. 1, pp. 1-14, October 2005.
- [55] A. Birk, and S. Carpin, "Merging occupancy grids from multiple robots", Proceedings of the IEEE, Vol. 94, No. 7, pp. 1384-1397, July 2006.
- [56] Gonzalez, R. C., & Woods, R. E., (1993). Digital Image Processing. Springer.
- [57] V. Nguyen, A. Martinelli, N. Tomatis and R. Siegwart, "A Comparison of Line Extraction Algorithms using 2D Laser Range Finder for Indoor Mobile Robotics", Conference on Intelligent Robots and Systems, IROS'2005, pp. 1929-1934, Edmonton, Canada, 2-6 August, 2005.
- [58] J. Neira and Juan D. Tardós, "Data Association in Stochastic Mapping using the Joint Compatibility Test". IEEE Transactions on Robotics and Automation, Vol.17, No. 6, pp. 890 - 897, December 2001.
- [59] A. A. Aghamohammadi, H. D. Taghirad, A. H. Tamjidi and E. Mihankhah, "Feature-Based Laser Scan Matching For Accurate and High Speed Mobile Robot Localization", Proceedings of the 3rd European Conference on Mobile Robots (ECMR'07), Freiburg, Germany, 19-21 September, 2007.
- [60] "Lejos, Java for Lego Mindstorms". Available at "<http://lejos.sourceforge.net/>", June 2012.
- [61] P. Costa, "Paco - Wiki: SimTwo". Available at "<http://paginas.fe.up.pt/~paco/wiki/index.php?n=Main.SimTwo>", June 2012.
- [62] H. Sobreira, F. Santos, H. Alves and A. P. Moreira, "Localizing an NXT Lego Robot using Infra-Red Beacons", EPIA2011 - 15th Portuguese Conference on Artificial Intelligence, Lisbon, Portugal, 10-13 October, 2011.
- [63] A. Behrens, L. Atorf, R. Schwann, B. Neumann, R. Schnitzler, J. Ballé, T. Herold, A. Telle, T. G. Noll, K. Hameyer and T. Aach, "MATLAB Meets LEGO Mindstorms — A Freshman Introduction Course Into Practical Engineering", IEEE Transactions on Education, Vol. 53, No. 2, pp. 306-317, May, 2010.
- [64] S. H. Kim and J. W. Jeon, "Introduction to Embedded Systems Using Lego Mindstorms", IEEE Transactions on Education, Vol. 52, No.1, pp. 99-108, February, 2009.
- [65] S. Sharad, "Introducing Embedded Design Concepts to Freshmen and Sophomore Engineering Students with LEGO MINDSTORMS NXT", IEEE International Conference on Microelectronic Systems Education, San Diego, California, USA, 3-4 June, 2007.
- [66] S. H. Kim and J. W. Jeon, "Educating C Language Using LEGO Mindstorms Robotic Invention System 2.0", IEEE International Conference on Robotics and Automation, Orlando, Florida, USA, 15-19 May, 2006.
- [67] Gonçalves, J., (2009), "Modelação e Simulação Realista de Sistemas no Domínio de Robótica Móvel", Thesis, (PhD), Faculty of Engineering of the University of Porto, Porto, Portugal.
- [68] M. Pinto, A. P. Moreira and A. Matos, "Localization of Mobile Robots Using an Extended Kalman Filter in a LEGO NXT", IEEE Transactions On Education, Vol- 55, No 1, pp. 135-144, February 2012.



- [69] Martin Riedmiller and Heinrich Braun, "A Direct Adaptive Method for Faster Back-Propagation Learning: the RPROP algorithm". IEEE International Conference on Neural Networks, pp. 586–591, San Francisco, California, USA, 28-March to 1-April, 1993.
- [70] Miguel Pinto, Miguel Pinto, A. Paulo Moreira, Aníbal Matos, Héber Sobreira, "Global Localization Algorithm from a Multiple Hypotheses Set". 9th Latin American Robotics Symposium – LARS, Fortaleza, Brazil, 15-21 October, 2012.
- [71] Miguel Pinto, A. Paulo Moreira, Aníbal Matos, Héber Sobreira, Filipe Santos, "Fast 3D Map Matching Localization Algorithm", International Conference on Computer and Automation Engineering (ICCAE 2013), Brussels, Belgium, 12-13 January, 2013.
- [72] Miguel Pinto, A. Paulo Moreira, Aníbal Matos, Héber Sobreira, "Novel 3D Matching Self-Localization algorithm", International Journal of Advances in Engineering and Technology, Vol. 5, No. 1, November, 2012.
- [73] "Robótica 2012, Festival Nacional". Available at "<http://www.robotica2012.org/12/>", June 2012.
- [74] "INESC TEC Porto". Available at "<http://www2.inescporto.pt/>", July, 2012.
- [75] "Fórum do Mar". Available at "<http://www.forumdomar.exponor.pt/>", July, 2012.
- [76] "ROS-Gmapping". Available at "<http://www.ros.org/wiki/gmapping>", November, 2012.
- [77] "ROS-Amcl". Available at "<http://www.ros.org/wiki/amcl>", November, 2012.